HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

BACHELORARBEIT

Herr
Michael Ohme

Live-Updates von Oracle-Linux-Servern mit Ksplice und Yum

Untersuchung der Systemstabilität

2015.

Fakultät **Angewandte Computer- und Biowissenschaften (CB)**

BACHELORARBEIT

Live-Updates von Oracle-Linux-Servern mit Ksplice und Yum

Untersuchung der Systemstabilität

Autor:
Michael Ohme

Studiengang:
Medieninformatik & Interaktives Entertainment

Seminargruppe:
MI11w2-B

Erstprüfer:
Prof. Dr.-Ing. Uwe Schneider

Zweitprüfer:
Dipl.-Inf. Thilo Solbrig

Mittweida, 30.11.2015.

Bibliografische Angaben

Ohme, Michael: Live-Updates von Oracle-Linux-Servern mit Ksplice und Yum, Untersuchung der Systemstabilität, 129 Seiten, 13 Abbildungen, 25 Tabellen, 15 Anlagen, Hochschule Mittweida, University of Applied Sciences, Fakultät Angewandte Computer- und Biowissenschaften (CB)

Bachelorarbeit, 2015.

Satz: LaTeX

Referat

Server müssen hohen Sicherheitsansprüchen genügen und dürfen in der Praxis so wenig wie möglich ausfallen. Dabei ist bereits die Zeit für eine Wartung ein hoher Kostenfaktor. Systemaktualisierungen sollten so schnell wie möglich eingespielt werden, am besten während des Produktivbetriebs. Diese Arbeit setzt hier an und untersucht die zwei Programme *Ksplice* und *Yum*. Diese Aktualisierungssysteme für *Oracle*-Linux sollen das Betriebssystem aktuell halten ohne es abschalten zu müssen.

Ziel dieser Arbeit ist es, eine Aussage darüber treffen zu können, ob mit den zwei Programmen Ausfallzeiten für Wartungsarbeiten minimiert werden können. Hierzu wird die Praxistauglichkeit der Programme erprobt und nach Schwächen gesucht, welche die Serverstabilität gefährden. Es werden die Funktionsweisen analysiert und die gewonnenen Erkenntnisse durch praktische Tests untermauert.

Danksagung

Zunächst möchte ich mich an dieser Stelle bei all denjenigen bedanken, die mich während der Anfertigung dieser Bachelorarbeit unterstützt und motiviert haben.

Ganz besonders gilt dieser Dank Herrn Prof. Uwe Schneider, der meine Arbeit und mich betreut hatte. Trotz seiner zahlreichen Verpflichtungen war er immer für mich zu erreichen und unterstützte meine Arbeit durch viele Ratschläge.

Ebenso danke ich Herrn Thilo Solbrig, welcher meine Arbeit in der Firma betreute. Durch sein großes Fachwissen konnten Probleme immer schnell aus der Welt geschaffen werden, was den Fortschritt der Arbeit begünstigte. Mein Dank gilt auch der Firma ASPICON GmbH, die mir die Möglichkeit gegeben hatte bei ihnen zu forschen und zu arbeiten.

Meinen Freunden Anna-Marie und Matthias danke ich auch sehr. Sie haben zahlreiche Stunden Korrektur gelesen, wiesen mich auf Schwächen hin und reflektierten mir kritisch meinen Arbeitsstand.

Zu guter Letzt bedanke ich mich ganz herzlich bei meiner Freundin Tina, die zahlreiche Kommata, Satzstellungen und Rechtschreibfehler korrigiert hat.

Inhaltsverzeichnis

Abbildungs- und Tabellenverzeichnis

Abbildungen

Tabellen

Abkürzungsverzeichnis

API ..... Application Programming Interface, Seite 23
ASM ..... Automatic Storage Management, Seite 14
Btrfs ..... B-tree File System, Seite 16
CMWQ ..... Concurrency Managed Workqueues, Seite 58
cpio ..... Copy in, Copy out, Seite 32
CRC ..... Cyclic Redundancy Check, Seite 33
CUPS ..... Common Unix Printing System, Seite 14
DB ..... Datenbank, Seite 17
DEB ..... Debian-Pakete, Seite 15
DLL ..... Dynamic Link Library, Seite 12
DSU ..... Dynamische Software Updates, Seite 18
GNU ..... GNU's Not Unix, Seite 13
GPG ..... GNU Privacy Guard, Seite 31
GPL ..... General Public License, Seite 16
GUI ..... Graphical User Interface, Seite 30
HTML ..... Hypertext Markup Language, Seite 23
HTTP ..... Hypertext Transfer Protocol, Seite 23
RAC ..... Real Application Clusters, Seite 17
RCK ..... Red Hat Compatible Kernel, Seite 16
REST ..... Representational State Transfer, Seite 23
RPM ..... RPM-Package-Manager, Seite 4
SVR4 ..... System Five Release 4, Seite 33
TPM ..... Transaktionen pro Minute, Seite 93
UEK ..... Unbreakable Enterprise Kernel, Seite 16
ULN ..... Unbreakable Linux Network, Seite 17
URI ..... Uniform Resource Identifier, Seite 23
VK ..... Verfügbarkeitsklassen, Seite 3
XFS ..... X-File System, Seite 16
YUM ..... Yellowdog Updater, Modified, Seite 4
Yup ..... Yellowdog Updater, Seite 29

1 Einleitung

Fast jedes Computersystem muss in regelmäßigen Abständen mit Verbesserungen in Form von Updates versorgt werden. Eine besondere Rolle nehmen dabei Serversysteme ein. Diese haben hohe Ansprüche an Sicherheit, Stabilität und Verfügbarkeit. Dabei stehen diese Ansprüche teilweise in Konflikt miteinander. Beispielsweise kann ein Server, der immer verfügbar sein soll, nicht für Wartungsarbeiten heruntergefahren werden. Für diese Herausforderungen gibt es Lösungen, die es ermöglichen Updates während der Laufzeit in das System einzuspielen. Diese Arbeit beschäftigt sich mit einer dieser Lösungen des „Live-Patchings“ und untersucht deren Stabilität.

1.1 Problemfelder bei Updates im Serverbereich

„Je komplexer ein System wird, desto fehlerhafter wird es sein.“[1]

Besonders deutlich wird dieses Zitat bei modernen Computersystemen, deren Komplexität zunehmend größer wird. Durch Verbesserungen in der Fertigungstechnik passt immer mehr Schaltlogik und andere Komponenten auf einen einzelnen Computerchip. So haben Prozessoren, die im 45-nm-Verfahren gefertigt werden, ca. 820 Millionen Transistoren. [2] Zusätzlich wird versucht, immer mehr Funktionen in einer Baugruppe zusammenzuschließen, beispielsweise Sound- und Netzwerkkarte, welche mit auf die Hauptplatine eingebunden werden. Bei Computersystemen kommt zu der Komplexität der Hardware auch noch die der Software hinzu. Der Linux-Kern besitzt in der Version 3.10 fast 17 Millionen Zeilen Code [3] und stellt dabei nur den Kern eines Betriebssystems dar. Je nach Aufgabe des Systems kommen zahlreiche Software und Dienste hinzu, die eine nicht unerhebliche Komplexität mit sich bringen. Allein die *MySQL*-Datenbank besteht aus ca. 2,5 Millionen Codezeilen. [4] Jede dieser Software übernimmt aber nur eine bestimmte Arbeit und muss daher direkt oder indirekt in Verbindung mit anderen Programmen stehen. Aber nicht nur Anwendungen auf einem einzelnen Computer interagieren miteinander oder konkurrieren um Ressourcen. Durch das Internet und die zunehmende Vernetzung von Computersystemen wächst der Informations- und Datenaustausch. Diese tragen ihren Teil zur Komplexität bei. Weiterhin sollte bedacht werden, dass jedes Stück Software und Hardware von einer anderen

[1] *Ahmad-Reza Sadeghi: Center for Advanced Security Research, Darmstadt, 2014.* [1]
[2] *Vgl. Intel Corporation (Hrsg.): Intel's Fundamental Advance in Transistor Design Extends Moore's Law, Computing Performance Sixteen Eco-Friendly, Faster and 'Cooler' Chips Incorporate 45nm Hafnium-Based High-k Metal Gate Transistors.* [2]
[3] *Vgl. Stephen Shankland: Linux development by the numbers: Big and getting bigger, 2013.* [3]
[4] *Vgl. Black Duck Software, Inc. (Hrsg.): MySQL.* [4]

Personengruppe entwickelt und gefertigt wird. Sei es ein Unternehmen, eine Einzelperson oder eine Open-Source-Gruppe. Dabei hat jede dieser Gruppen einen eigenen Anspruch an Qualität und Komplexität ihres Produktes. Wer sich dieser und weiterer Einflussfaktoren eines Computersystems bewusst ist, kann nachvollziehen, dass Software stetig verbessert werden muss. Dabei wird sie von Fehlern befreit, ihre Kompatibilität zu anderen Programmen oder der Hardware erhöht oder ihr Funktionsumfang erweitert. Diese Verbesserungen werden in Form von Updates durchgeführt und in Abhängigkeit ihres Schwerpunktes unterschiedlich benannt, z.B. Hotfix, Update oder Patch.

Der verbesserte Quellcode wird dabei meist über ein Netzwerk zu dem Computer gesendet, welcher das Update angefordert hat. Dort ersetzt der neue Binärcode die alten Daten auf der Festplatte. Die Neuerungen müssen noch in den Arbeitsspeicher geladen werden, um sie zu verwenden. Dies wird üblicherweise mit einem Neustart des Programms erreicht oder mit dem neue Laden einer Bibliothek. Bei einem Musikplayer z.B. ist dies innerhalb von Sekunden erledigt und auch nicht kritisch. Auf die Musik kann ein paar Sekunden verzichtet werden und es gibt keine Abhängigkeiten zu anderen wichtigen Programmen. Bei anderen Anwendungen kann diese Arbeitsunterbrechung umfangreichere Folgen haben. Beispielweise kann der 30-sekündige Ausfall einer Datenbank die darauf aufbauenden Programme stark beeinflussen, vor allem, wenn diese nicht gut programmiert sind und den kurzen Verbindungsabbruch nicht kompensieren können. Ähnlich verhält es sich bei Diensten. Diese laufen im Hintergrund eines Systems und stellen ihre Funktionalitäten anderen Programmen zur Verfügung. Wenn diese ausfallen, können andere Programme nur eingeschränkt bis gar nicht betrieben werden. Unter gewissen Umständen kann es vorkommen, dass alle voneinander abhängigen Programme neu gestartet werden müssen.

Bei einem Betriebssystem sind die Folgen eines Ausfalls noch weitreichender. Es benötigt nicht nur länger für den Neustart, sondern stellt zudem grundlegende Programmbibliotheken zur Verfügung, von denen andere Programme abhängig sind. Weiterhin verwaltet es die Hardwareressourcen und ist unter anderem für sicherheitskritische Abläufe verantwortlich. Daher ist der Neustart eines Betriebssystems wesentlich kritischer. Wenn es sich bei dem System noch um einen Server handelt, der bestimmte Dienste permanent zur Verfügung stellen soll, kann ein einfacher Neustart zu einer Herausforderung werden. Dabei geht es nicht nur um die Tatsache, dass der vom Server angebotene Dienst nicht mehr zur Verfügung steht, sondern auch um die Wiederanlaufzeit der Anwendungen, die auf dem Serverdienst aufbauen. Das Starten aller Programme, Dienste und von diesen wiederum abhängigen Dienste und Programme kann mehrere Stunden betragen.

„Je mehr die Geschäftsprozesse von der IT abhängen, desto seltener werden Fälle, in denen der Stillstand des IT- Systems nur geringfügige Folgen hat.“ [5] Deutlich wird das, wenn die Kosten dafür betrachtet werden. So können im Einzelhandel die Kosten für eine Stunde Downtime ca. 90.000 € betragen. Bei einem Rechenzentrum, beispielsweise

[5] *managedhosting.de GmbH (Hrsg.): Ausfallsicherheit von IT-Systemen, Version 2.0, S2.* [5]

im Bankensektor, sind es schon 2,5 Millionen € pro Stunde. [6] Daher legen viele Unternehmen regelmäßige Termine für derartige Wartungen fest, auch „Patchdays“ genannt. Diese werden an arbeitsschwachen Zeitpunkten, wie dem Wochenende und in den Abendstunden, durchgeführt. Je nach Umfang kann das ein knappes Zeitfenster für den Administrator sein. Zudem sind es ungeliebte Arbeitszeiten, die erhöhte Kosten zur Folge haben.

„Patchdays“ haben auch soziale Konsequenzen. Die verantwortlichen Personen müssen ihr Privatleben hinter das berufliche stellen. Besonders deutlich wird dies bei IT-Dienstleistungsunternehmen. Ein Administrator ist meist für mehrere Kunden und deren Server verantwortlich, dabei können Verhältnisse von 20 Kunden pro Betreuer entstehen. Selbst wenn diese Unternehmen ihre „Patchdays“ nicht auf das gleiche Datum legen, so sinkt die Anzahl der freien Wochenenden für den Administrator. Eine weitere Herausforderung ist es, wenn zu dem „Patchday“ von Kunde A noch der Notfallsupport von Kunde B kommt. Für solche Fälle werden Bereitschaftsdienste eingeteilt, bei denen ein Administrator seine Freizeit unterbricht und das Unternehmen unterstützt. Auch wenn dabei der Mitarbeiter nur im Notfall arbeiten muss, wird die Gestaltung seiner privaten Zeit eingeengt.

Immer häufiger gibt es die Anforderung von „hochverfügbaren Diensten“, d.h Dienste die pro Jahr nur eine bestimmte Zeit ausfallen dürfen. Diese Dienste werden von Programmen zur Verfügung gestellt, welche auf einem Server betrieben werden. Je nach minimaler Zeit, die ein Server nicht erreichbar sein darf, werden diese in Verfügbarkeitsklassen (VK) unterteilt. Beispielsweise darf ein Server der Verfügbarkeitsklasse 3 nicht mehr als 53 Minuten im Jahr ausfallen. Diese Anforderung steigert sich über die „Höchstverfügbarkeit“ (VK 4, mehr als 6 min pro Jahr) hin zu „Desaster-Tolerant“ (VK 5, gar keine Ausfallzeit). Eine Übersicht über die einzelnen Verfügbarkeitsklassen befindet sich im Anhang, in der Tabelle F.1.

Immer mehr Unternehmen haben solche Anforderungen, vor allem jene, deren kritische Geschäftsprozesse auf den Betrieb von Servern angewiesen sind. Damit sind nicht nur Onlineshops, Banken und Serverfarmen gemeint. Alle Unternehmen, die zunehmend papierlos arbeiten, benötigen permanenten Zugriff auf ihre Daten. Beispielsweise verwaltet „Ein Krankenhaus [...] im Durchschnitt 800.000 Patientenakten, eine 3D-Computertomografie allein beansprucht im Durchschnitt 1 Gigabyte Speicher.“ [7] „Während Dienstleistungs- oder Produktionsbetriebe Ausfälle ggf. mit Mehrarbeit kompensieren können, sind Leistungserbringer aufgrund möglicher fataler Folgen für die Gesundheit ihrer Patienten zwingend auf eine Hochverfügbarkeit ihrer Anwendungen und Daten angewiesen.“ [8]

[6] *managedhosting.de GmbH (Hrsg.): Ausfallsicherheit von IT-Systemen, Version 2.0, S2.* [5]
[7] *DataCore (Hrsg.): Aktuelle IT-Anforderungen im Gesundheitswesen, 2015.* [7]
[8] *Wiley-VCH Verlag GmbH & Co. KGaA (Hrsg.): Hochverfügbarkeit der IT-Infrastruktur wichtig für Patientensicherheit, 2007.* [8]

Dies sind einige Gründe, warum IT-Dienstleister und entsprechende Abteilungen in Unternehmen motiviert sind, ihre Serversysteme so wenig wie möglich abzuschalten und die Patchdays in recht großen Zeitabständen durchzuführen. Um so interessanter und nützlicher ist es, wenn die Updates während der Laufzeit des Betriebssystems eingespielt werden und ein Neustart des Systems nicht notwendig wäre.

1.2 Aktualisierungen während der Systemlaufzeit

Die 2008 gegründete Firma *Ksplice* [9] entwickelte eine Software für Linux-Systeme, mit der es möglich ist den Linux-Kern zu aktualisieren, ohne das ein Neustart nötig ist. Diese wurde später von dem Unternehmen *Oracle* gekauft, welches die Software unter dem Namen *Ksplice* für Kunden mit einem *Oracle Linux Premier Support* bereitstellt. [10] Die Funktionen dieser Software beschränken sich ausschließlich auf den Linux-Kern und aktualisieren keine anderen andere Komponenten des Systems, wie Bibliotheken oder Programmdaten.

An dieser Stelle gibt es noch Handlungsbedarf, denn ein System ist nur dann gut geschützt und dessen Stabilität kann gewährleistet werden, wenn alle Softwarekomponenten aktualisiert werden. Dies wird unter Linux meist mit dem Paketmanager des jeweiligen Systems durchgeführt. Es ist nicht bekannt, dass ein Paketmanager existiert, der es erlaubt Anwendungen während ihrer Laufzeit zu aktualisieren. Es fehlen also die Mittel, um auch andere Teile des Betriebssystems auf dem neusten Stand zu halten, ohne das System oder einzelne Anwendungen neu starten zu müssen. Vor allem Computersysteme mit einer hohen Verfügbarkeitsklasse benötigen hierfür geeignete Lösungen. Bei diesen muss vor Inbetriebnahme geprüft werden, ob der Funktionsumfang des Paketmanagers ausreicht, um die restlichen Komponenten des Systems, entsprechend der Verfügbarkeitsklasse, aktuell zu halten. Zudem sollte das Einspielen von “Kernel-Live-Patches” das System nicht destabilisieren.

1.3 Untersuchung von Update-Systemen

Ziel dieser Arbeit ist die Praxistauglichkeit eines Live-Patching-Systems zu untersuchen. Dabei wird der Fokus auf die Stabilität des Systems und die Vermeidung von Downtime gesetzt, welche durch die Wartung des Betriebssystems entstehen können. Dies wird am Beispiel von *Oracle-Linux* und der dafür zertifizierten Software *Ksplice* getan. Unter dem gleichen Gesichtspunkt wird der zur Distribution gehörende Paketmanager *Yum* (Yellowdog Updater, Modified) bzw. *RPM* (RPM-Package-Manager) untersucht. Es wird im Folgenden nicht auf alle Systeme eingegangen, auf denen *Ksplice* und *Yum* betrieben werden können. Dieser Arbeit beschränkt sich auf den Einsatz der Software unter der

[9] *Vgl. CrunchBase (Hrsg.): Ksplice.* [9]
[10] *Vgl. Thoma, Jörg: Oracle schließt Red Hat und Suse vom Support aus, 2011.* [10]

Distribution *Oracle-Linux*. So soll folgendes geklärt werden:
Sind *Ksplice* und *Yum* geeignete Werkzeuge, um Downtimes im Produktiveinsatz auf *Oracle*-Datenbank-Servern zu reduzieren? Der Verringerungsgrad soll dabei so hoch sein, dass zwischen Patchdays gar keine Downtime mehr nötig ist.

1.4 Partnerunternehmen der Abschlussarbeit

Diese Arbeit wurde von der Firma ASPICON GmbH betreut. Das in Chemnitz ansässige Unternehmen ist im Bereich der IT-Dienstleitungen tätig. Das Unternehmen wurde 2002 gegründet und spezialisierte sich seit seiner Gründung auf Produkte von *Oracle*. Es gilt als kleinster „Oracle Certified Advantage Partner“ in Deutschland und hält den Status als „Oracle Platinum Partner“. Im Kundenbereich setzt ASPICON keinen Branchenschwerpunkt und unterstützt über 500 Firmen im In- und Ausland.

Als spezialisierter Experte bietet das Unternehmen vielseitige Dienste im Bereich der Administration von Servern, Cluster, virtuellen Maschinen und Datenbanken an. Unter anderem entwickelt es mit Kunden und Partnern Konzepte für hochverfügbare IT-Infrastrukturen. Zu den Dienstleitungen der ASPICON GmbH gehört neben der Administration auch das Überwachen von Kundensystemen, Schulungen, Lizenzverwaltung und Softwareentwicklung.

Die Mitarbeiter des Unternehmens haben sich auf Datenbanklösungen von *Oracle* und *MySQL* spezialisiert. Im Bereich der Betriebssysteme liegt der Schwerpunkt bei *Oracle-Enterprise-Linux*, es werden aber auch andere Distributionen, wie *Windows* und *Solaris*, betreut. Im Bereich der Virtualisierung hält das Unternehmen für die Produkte von *VmWare* und *Virtual-Box* Expertenwissen vorrätig.

Das Unternehmen ist besonders auf die Stabilität, Sicherheit und Verfügbarkeit von Servern bedacht. Konflikte zwischen diesen Zielen sind nicht selten. Daher ist es interessant zu untersuchen, wie gut es mit Produkten von *Oracle* möglich ist, diese drei Ziele gleichermaßen gut zu erreichen. [11]

1.5 Struktur und Methodik der Arbeit

Der Autor geht mit einer induktiven Methodik an die Problemstellung heran. Das Thema der Arbeit legt das bereits nahe. Dieses spezialisiert die Untersuchungen auf ausgewählte Software. So wird nicht eine Vielzahl an Linux-Distributionen oder Paketmanager untersucht, sondern nur speziell festgelegte.

Nach der Einleitung und Formulierung der zentralen Frage folgt ein Kapitel, welches in

[11] *Vgl. ASPICON GmbH (Hrsg.):Firmenportrait, Chemnitz 2015.* [11]

Begrifflichkeiten einführt und Hintergrundwissen vermittelt, was für das Verständnis der Arbeit hilfreich ist. Der restliche Teil der Arbeit ist in zwei größere Abschnitte gegliedert. Der erste besteht aus den Kapiteln drei und vier. In ihnen wird die zu untersuchende Software theoretisch analysiert. Unter Zuhilfenahme der Literaturrecherche werden die Untersuchungsobjekte vorgestellt, ihre Funktionsweise beschrieben und Informationen zur Fragestellung zusammengetragen. Der Autor betrachtet das als eine Form der statischen Analyse, da die zu analysierenden Programme nicht ausgeführt werden. [12]

Der zweite Teil ist eine dynamische Analyse und wird in Kapitel 5 näher beschrieben. [13] Durch empirische Tests werden die Sachverhalte untersucht, welche die statische Analyse nicht beantworten konnte. So arbeitet sich der Autor über konkrete Fakten an die Beantwortung der zentralen Fragestellung heran. Die induktive Methodik erlaubt es von dem Besonderen auf das Allgemeine zu schließen. Dies wird vor allem in den praktischen Untersuchungen deutlich. Die darin analysierte Software wird auf vielen anderen Computersystemen verwendet. *Oracle*-Linux setzt dabei immer den Paketmanager *Yum* zum Aktualisieren der Software ein. Die Software *Ksplice* wird sogar nur von *Oracle* vertrieben. So kann von den Testsystemen auch auf andere Oracle-Linux-Systeme geschlossen werden, welche die gleiche Software einsetzen. [14] Dieses Vorgehen wurde auch aus pragmatischen Gründen gewählt. Denn es ist in der vorgegebenen Zeit nicht möglich, eine Vielzahl von Computersystemen zu testen.

[12] *Vgl. Hardt, Dennis: Werkzeuggestützte Softwareprüfung Statische Analyse und Metrik, S. 3, S. 6f, Hannover 2006.* [12]

[13] *Vgl. Koschke, Rainer: Vorlesung Software Reengineering, S. 3ff, Bremen 2010.* [13]

[14] *Vgl. Krieglsteiner, Susann: Wissenschaft und Fachtheorie. Methoden und Techniken der Disziplin, S. 144.* [14]

2 Grundlagen zu Oracle, Linux und Aktualisierungen

Dieses Kapitel dient zur Einführung in die Thematik und erleichtert das Lesen der folgenden Kapitel. Im ersten Abschnitt werden Informationen vermittelt, die zum allgemeinen Verständnis der Arbeit nützlich sind. Das erste Unterkapitel vermittel grundlegende Kenntnisse zu Linux und geht auf Betriebssystemkomponenten ein, die für diese Arbeit relevant sind. Darauf folgt ein Abschnitt über Software von Oracle, die in den praktischen Untersuchungen eingesetzt wird. Unter anderem wird die Datenbank und die darunter liegende Software kurz erläutert. Im abschließenden Unterkapitel wird auf die, zum Verfassungszeitpunkt dieser Arbeit, aktuelle verbreitete Technik eingegangen.

2.1 Abgrenzung von Update-Begriffen

Im heutigen Sprachgebrauch werden Begriffe wie Bugfix, Update und Service-Pack meist gleichbedeutend verwendet, obwohl es zwischen ihnen deutliche Unterschiede gibt. Im Folgenden wird auf die einzelnen Begrifflichkeiten eingegangen, um sie besser voneinander abgrenzen zu können.

- Update:
  Unter einem Update wird im Sinne der Informatik eine Erweiterung und Verbesserung einer Software verstanden. In der Brockhaus-Enzyklopädie wird es als „[...] jede Aktualisierung eines Informations- bzw. Datenträgers, [...]“ [15] definiert. „Dabei fügt ein U. dem bisherigen Produkt z.T. geringfügige neue Funktionen hinzu oder korrigiert Fehler, die nach der Veröffentlichung entdeckt wurden.“ [15]

- Upgrade:
  Der Begriff des Upgrades kann im Allgemeinen synonym für ein Update verwendet werden, da es keine genaue Definition hierfür gibt. Die Brockhaus-Enzyklopädie bezeichnet die Differenzierung zwischen Update und Upgrade als, „ [...] z.T. unscharf [...] “. [15] Im Linux-Umfeld werden die beiden Begriffe sehr unterschiedlich verwendet. In dem Paketmanager *APT* beschreibt der Befehl `apt-get update` das Aktualisieren der lokalen Paketliste vom Repository-Server und mit `apt-get upgrade` werden die installierten Programme aktualisiert. [16] Im Gegensatz dazu aktualisiert der Paketmanager *Yum* mit dem Befehl `yum update` die installierten

[15] *Brockhaus Enzyklopädie in 12 Bänden: Band 11 STIN-VERD, 21. Auflage, Update S. 609, Leipzig / Mannheim 2005.* [15]

[16] *Vgl. Ubuntu Deutschland e.V. (Hrsg.): Wiki, apt-get, 2015.* [16]

Programme.

- Service-Pack:
  Dieser Begriff wurde von der Firma *Microsoft* geprägt und beschreibt eine Zusammenfassung mehrerer Aktualisierungen und Verbesserungen für das Betriebssystem und andere *Microsoft*-Produkte. *Microsoft* selbst beschreibt dies als „[...] ein Windows-Update, in dem häufig bereits veröffentlichte Updates kombiniert werden, um die Zuverlässigkeit von *Windows* zu erhöhen. Service Packs [...] können Sicherheits- und Leistungsverbesserungen sowie Unterstützung für neue Typen von Hardware enthalten.“ [17]

- Patch:
  Dieser Begriff stammt aus dem Englischen und bedeutet übersetzt „Flicken“ oder „Ausbessern“. Als Patch werden Aktualisierungen bezeichnet, die kleine Fehler beheben, teilweise ergänzen sie auch kleinere Funktionen einer Software. Der Begriff kann in abgewandelter Form genutzt werden um eine spezielle Verwendung zu beschreiben. So wird ein Patch der vorwiegend eine Sicherheitslücke schließt als Sicherheitspatch bezeichnet. [18]

- Bugfix:
  Ein Bugfix ist eine kleinere Aktualisierung und behebt einen oder mehrere Fehler in einem Programm. Es werden in der Regel keine neuen Funktionen ergänzt. [19]

- Hotfix:
  Ein Hotfix ist eine spezieller Bugfix, der einen konkreten Fehler möglichst schnell beheben soll. Der Begriff setzt sich aus den englischen Wörtern „Hot-“ für „heiß“ und „-fix“ für „reparieren“ zusammen. Ein Hotfix ist eine reine Fehlerkorrektur und enthält keine neuen Funktionen. Durch die hohe Dringlichkeit, mit der ein Hotfix erstellt wird, ist er seitens des Herstellers weniger intensiv getestet. Meist wird er nur auf Grund einer konkreten Supportanfrage des Kunden erstellt und bezieht sich auf spezielle Kundensituationen und Kundenprobleme. [18]

- Kernel-Live-Patching:
  Bezeichnet den Vorgang, bei dem Aktualisierungen während der Laufzeit des Betribssystemkerns durchgeführt werden und kein Neustart des Computersystems nötig ist, um den geänderten Binärcode zu verwenden. Die Aktualisierungsdatei selbst wird „Live-Patch“ genannt und behebt kleine bis mittelgroße Probleme oder enthält Optimierungen. In dieser Arbeit wird der Begriff Live-Patching auch für die

[17] *Vgl. Microsoft Corporation (Hrsg.):Service Pack und Update Center, 2015.* [17]
[18] *Vgl. Bundesamt für Sicherheit in der Informationstechnik (Hrsg.): Maßnahmenkatalog, M 3.66 Grundbegriffe des Patch- und Änderungsmanagements, 2008.* [18]
[19] *Vgl. Sneed, Harry u.a.: Software- Produktmanagement, Wartung und Weiterentwicklung bestehender Anwendungssysteme, S. 88-92, 1. Auflage, Heidelberg 2005.* [19]

Aktualisierungsvorgänge genutzt, welche sich nicht auf den Betriebssystemkern beschränken, aber während der Programmausführung eingespielt werden. [20]

Anhand der Ausführung wurde geklärt, in welcher Form eine Aktualisierung vorkommen kann, aber noch nicht wie diese verteilt werden. Ein umständlicher Weg ist es, die Aktualisierung auf einem Datenträger, wie einem USB-Stick oder einer CD zum Zielcomputer zu transportieren und dort einzuspielen. Ein komfortablerer Weg ist es, die Aktualisierung über ein Netzwerk bis zum Zielsystem zu übertragen. Dafür müssen der Server, der die Aktualisierung vorrätig hält und der Client, der diese installieren möchte, miteinander kommunizieren. Dafür existieren zwei Modelle: Push und Pull.

- Pull-Modell (engl. Ziehen):
  Bei diesem Vorgang ist der Informations-Empfänger (consumer) der Akteur, bzw. Entscheidungsträger. Er ist in der „Holschuld“ und muss aktiv werden und über den Übertagungskanal die Information beim Produzent abholen. Im Sinne von Software-Aktualisierungen muss der Client beim Server immer nachfragen, ob ein Update verfügbar ist und es bei Bedarf abholen. Der Sender stellt die Daten an einer auffindbaren Stelle bereit, wo diese vom Sender gezogen (pull) werden können. [21]
- Push-Modell (engl. Drücken):
  Im Push-Modell agiert der Sender (supplier), d.h. er wählt bewusst konkrete Empfänger aus, an die er seine Informationen direkt sendet (pusht). Ein Beispiel ist das Pflichtupdate, was der Updateserver an eine Auswahl von Clients schickt. [21]

In der Regel liegen Aktualisierungen auf einem zentralen Server und der Client lädt diese von dort herunter. Die Entscheidung, wann und vor allem welche Daten übermittelt werden, liegt also auf der Seite des Client. Daher werden Aktualisierungen meist im Pull-Modell durchgeführt. Auch wenn der Nutzer an seinem Client Einstellungen vornimmt, die das System anweisen, Updates immer herunter zu laden und zu installieren, ist diese Entscheidung vom Client ausgegangen und wird damit als „Pull“ bezeichnet. [22]

2.2 Definition der Systemstabilität

Es existieren verschiedene Definitionen von Stabilität. Im Allgemeinen versteht man darunter „[...] die Eigenschaft eines Systems, seinen Zustand nicht oder kaum zu ändern.“ [23] In dieser Arbeit wird die Stabilität eines Betriebssystems fokussiert. Bei einem

[20] *Vgl. Kernel Live Patching, in: Wikipedia, Die freie Enzyklopädie, 2015.* [20]

[21] *Vgl. Oechsel, Rainer: Verteilte Systeme und Entwicklung verteilter Anwendungen, in: Schneider, Uwe / Werner, Dieter (Hrsg.): Taschenbuch der Informatik, 6. Auflage, S. 419, München 2007.* [21]

[22] *Vgl. Schäfer, Sebastian: Schaeferblick Weblog, Push vs. Pull in der Informationsverteilung – eine Definitionsfrage?, 2009.* [22]

[23] *Brockhaus Enzyklopädie in 12 Bänden: Band FARE-STIM, 21. Auflage, Stabilität S. 618, Leipzig / Mannheim 2005.* [23]

Betriebssystem spricht man von Stabilität, wenn es trotz äußerer Anregungen nicht abstürzt, in angemessener Zeit reagiert und kein Datenverlust entsteht. Durch verschiedene Faktoren wird diese Stabilität gefährdet. Dazu zählen andere Software (inklusive Treiber), fehlerhafte Hardware, Fehler im Betriebssystem selbst und Angriffe von außen. Diese „Destabilisatoren“ können fast alle durch Updates verringert werden. Dabei kann wiederum ein Update selbst zur Destabilisierung beitragen, da sich über diesen Weg Fehler in das System einschleichen können.

Es muss sichergestellt werden, dass ein System nicht durch Aktualisierungen (Updates) destabilisiert wird. Aus der Sicht des Autors wird dies am besten durch den Begriff der stabilen Eigenschaft ausgedrückt. Die Definition für eine stabile Eigenschaft E lautet: „Falls E für einen globalen Zustand Z zutrifft, so trifft E auch für den Folgezustand Z' (und damit für alle weiteren Folgezustände) zu.“ [24] Anders ausgedrückt: Ein Server hat die Eigenschaft E: „funktioniert fehlerfrei“ für den Zustand Z. Dieser Zustand gilt vor dem Einspielen eines Updates und bezieht sich auf installierte Programme und deren Versionen. Wenn E nach einer Aktualisierung (Update) des Systems, also im Zustand Z', auch noch gilt, dann ist E eine stabile Eigenschaft. Folglich kann die Stabilität eines Computersystems sichergestellt werden, indem es mit Updates versorgt wird, welche die stabile Eigenschaft E: „Betriebssystem ist stabil“ nicht verletzt.

2.3 Programmbibliotheken

In der Programmierung werden nützliche und oft benötigte Funktionen in separate Dateien ausgelagert. Alle Programme, die diese Funktionen benötigen, können diese gemeinsam über die ausgelagerte Datei nutzen. Solche Dateien werden Bibliotheken (eng. Libs) genannt. Sie beinhalten Programmfunktionen und statische Variablen. Es gibt unterschiedliche Bibliotheken. Je nachdem, welche Funktionen sie anbieten, können sie z.B. in grafische und mathematische Bibliotheken unterteilt werden. Als Systembibliotheken werden jene bezeichnet, die einer Anwendung Zugriff auf Betriebssystemfunktionen geben. Bibliotheken werden entsprechend nach ihrer Verwendung an unterschiedlichen Stellen im System gespeichert. Im Pfad `/lib` befinden sich unter Linux alle Bibliotheken, die für den Systemstart benötigt werden. Alle anderen zum System gehörenden sind im Verzeichnis `/usr/lib` zu finden. Die von einem Nutzer selbst hinzugefügten befinden sich meistens im Pfad `/usr/local/lib`. [25] Um Bibliotheken nutzen zu können, müssen sie an den Programmcode angebunden werden, dies wird auch „linken“ genannt. Abhängig vom Zeitpunkt des Bindungsvorgangs werden sie in statische Bibliotheken (static library), gemeinsame Bibliotheken (shared library) und dynamisch ladbare Bibliotheken (dynamic loadable library) kategorisiert.

[24] *Oechsel, Rainer: Verteilte Systeme und Entwicklung verteilter Anwendungen, in: Schneider, Uwe / Werner, Dieter (Hrsg.): Taschenbuch der Informatik, 6. Auflage, S. 436, München 2007.* [24]

[25] *Vgl. Dettmer, Steffen / Hemm, Torsten: SelfLinux, Bibliotheken, Version 0.12.1.* [25]

Die Benennung einer Bibliothek beginnt unter Linux immer mit dem Präfix `lib`, gefolgt von einem Namen, den der Entwickler festlegt. Bei statischen Bibliotheken wird ein „a“ (archiv) angehängt, was mit einem Punkt separiert wird. Bei dynamischen Bibliotheken folgt ein „so“ (shared object). Zum Schluss wird die Version angegeben. Daraus ergibt sich aber nicht der eigentliche Dateiname, sondern lediglich der `soname`. Gemeinsame Bibliotheken und dynamisch ladbare Bibliotheken sind sich sehr ähnlich. Namentlich gibt es zwischen ihnen keinen Unterschied, lediglich der Zeitpunkt des Bindens ist verschieden. In Abbildung 2.1 wird die Benennung nochmal verdeutlicht.

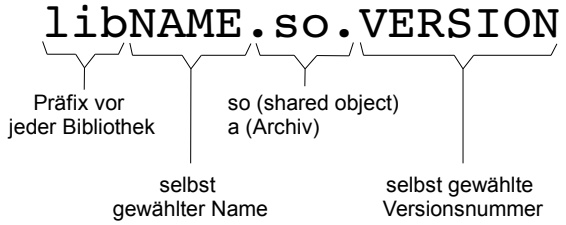

Abbildung 2.1: Benennung von Bibliotheken unter Linux [28]

Um eine gewisse Flexibilität bei der Versionsverwaltung zu erreichen, wird eine Bibliothek über mehrere Namen verlinkt. Mit ihrem `linker name` wird sie an das Programm gebunden. Dieser entspricht dem `soname` ohne Versionsnummer. Diese beiden werden durch eine symbolische Verknüpfung verbunden. Der eigentliche Dateiname der Bibliothek besteht aus dem `soname`, ergänzt um eine Release-Nummer. Diese Datei enthält auch den eigentlichen Binärcode und ist an den `soname` gebunden, siehe Abbildung 2.2. Die Verlinkungen können durch Installationsskripte vorgenommen werden, aber meistens sind die verlinkten Dateien schon in den Installationspaketen enthalten. [26]

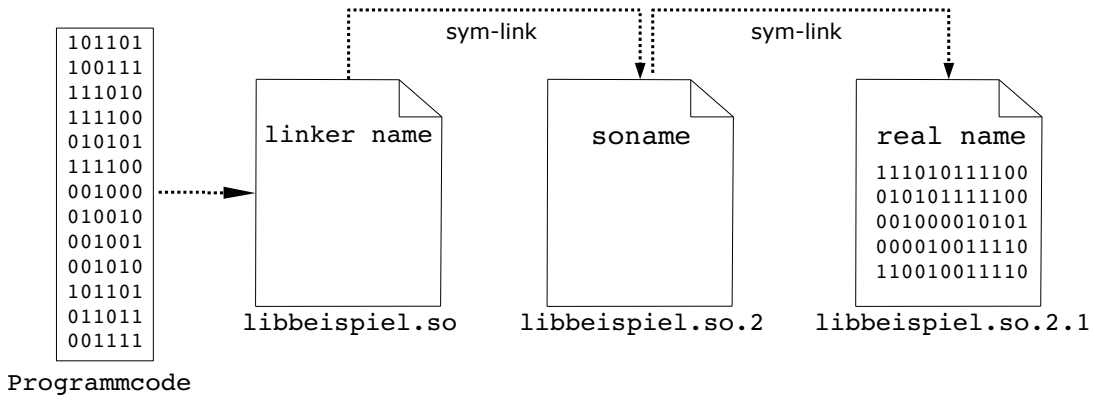

Abbildung 2.2: Verlinkung von Bibliotheken unter Linux [28]

Statische Bibliotheken (static library) werden beim Kompilieren an die Programme gebunden. Dabei wird die Bibliothek kopiert und zu einem Teil des Programms. Ein großer Vorteil ist, dass die Bibliothek nicht auf dem System vorhanden sein muss, um das

[26] *Wolf, Jürgen: Linux-Unix-Programmierung. Das umfassende Handbuch, 3.Auflage, S.860 , Bonn 2009.* [26]

Programm auszuführen. Dem Entgegen steht der höhere Verbrauch von Speicherplatz, sowohl auf der Festplatte/ Solid-State-Drive als auch im Arbeitsspeicher. Wenn beispielsweise der Browser *Firefox* unter Nutzung von statischen Bibliotheken mehrfach gestartet wird, dann werden auch die Bibliotheksfunktionen mehrfach in den Arbeitsspeicher geladen. Statische Bibliotheken können vom Anwender nicht aktualisiert werden, da diese fest im Programm integriert sind. Änderungen in einer statischen Bibliothek werden mit dem Programm aktualisiert.

Gemeinsame Bibliotheken (shared library) werden unter Linux als *shared object* (so) und unter *Windows* als *Dynamic Link Library*(DLL) bezeichnet. Sie werden vor dem Start eines Programms an den Programmcode gelinkt. Dafür prüft das System vor dem Programmstart, ob alle externen Referenzen aufgelöst werden können. Sollte eine Bibliothek fehlen, bzw. nicht in der richtigen Version vorhanden sein, wird der Programmstart mit einem Fehler abgebrochen. Andernfalls werden diese in den Arbeitsspeicher geladen, dort existiert jede Bibliothek nur einmal. Sollte diese bereits von einem anderen Programm genutzt werden und daher schon im Speicher liegen, wird sie lediglich an den Adressraum des Programms angebunden. Der Datenbereich der Bibliothek ist prozessspezifisch d.h., die Werte der Variablen werden für jedes Programm (Prozess) separat gespeichert. Der Programmcode existiert zur Laufzeit nur einmal im Arbeitsspeicher, der Datenbereich jedoch für jedes Programm, das die Bibliothek nutzt. Somit wird Redundanz vermieden, denn jedes Programm greift auf den gleichen Bereich im Arbeitsspeicher zu und die Bibliothek wird nicht unnötig oft geladen. Wenn der Arbeitsspeicher sich dadurch weniger aufbläht, müssen auch weniger Speicherseiten ein und ausgelagert werden (paging), was einen Geschwindigkeitsvorteil mit sich bringt. [27] Besonders positiv wirkt sich das bei Bibliotheken aus, die von vielen Programmen genutzt werden, wie den Systembibliotheken. [28]

Dynamisch ladbare Bibliotheken (dynamical loadable library) funktionieren ähnlich wie Gemeinsame Bibliotheken. Sie werden aber nicht beim Programmstart, sondern während der Laufzeit des Programms geladen. Die Nutzung einer gemeinsamen Bibliothek als dynamisch ladbare Bibliothek stellt höhere Anforderungen an die Programmierung der Anwendung. Dabei unterstützen verschiedene Systemfunktionen den Programmierer, wie z.B. `dlopen()` zum Laden und `dlclose()` zum Entladen einer Bibliothek. Im folgenden Code-Beispiel[28] wird die Verwendung von `dlopen()` gezeigt.

[27] *Vgl. Wladislaw, Eckhardt: Konzepte von Betriebssystem-Komponenten, Programmstart & dynamische Bibliotheken, 2005.* [27]

[28] *Vgl. Glatz, Eduard: Betriebssysteme. Grundlagen, Konzepte, Systemprogrammierung, S. 594 ff, Heidelberg 2006.* [28]

```
int main ()
  {
    void *h;                          // Handle für Library
    int (*convert)(int);              // Funktionszeiger auf conv
    h = dlopen("/lib/libmylib.so.2", RTLD_LAZY); // Lade mylib
    convert = dlsym(h, "conv");       // Setze Funktionszeiger
    return (*convert)(5);             // Rufe Bibliotheksfunktion
  }
```

„Der erste Aufrufparameter von dlopen() ist der Pfadname der Bibliothek. Der zweite Aufrufparameter ist eine Option die wir mit RTLD_LAZY so setzen, dass Externreferenzen der Bibliothek erst dann aufgelöst werden, wenn sie tatsächlich benötigt werden.“ [29]

2.4 Linux und relevante Systemkomponenten

Streng genommen ist Linux eine Software. Es stellt in seiner Funktion den Kern eines Betriebssystems dar und damit die Brücke zwischen der Hardware, dem Nutzer und der restlichen Software. Seine Entwicklung begann 1991 mit dem finnischen Studenten „Linus Torvalds“. Dieser stellte seinen Quellcode unter der GNU-Lizenz (GNU's Not Unix-Lizenz) zur Verfügung, so dass dieser von allen Interessenten genutzt und weiterentwickelt werden konnte. Das Besondere an der GNU-Lizenz ist, dass der veränderte und weiterentwickelte Code auch veröffentlicht werden muss und anderen Personen unter den gleichen Bedingungen wieder zur Verfügung steht. Linux unterliegt einer stetigen Weiterentwicklung unter der Leitung seines Erfinders. Das Betriebssystem wurde auf eine Vielzahl von Hardware portiert und kann auf unterschiedlichsten Rechnerarchitekturen für Desktop-Computer und Servern betrieben werden. Seine Vielfältigkeit und Anpassbarkeit demonstriert Linux auf verschiedenen Spielekonsolen wie der *Playstation*, *XBox* und *SteamBox*, aber auch auf mobilen Endgeräten wie in *Android* oder *Ubuntu-Touch*.

2.4.1 Linux Distributionen

Linux kann wie ein guter und leistungsstarker Motor gesehen werden. Ohne eine Karosserie, Bremsen oder Räder kann aber trotzdem nicht gefahren werden. Denn der Endbenutzer agiert mit dem Betriebssystem meist indirekt und arbeitet mit Anwender-Software auf einer grafischen Benutzeroberfläche, was Linux in seiner Grundform nicht bietet. Verschiedene Gruppen und Firmen haben auf der Grundlage von Linux Betriebssysteme zusammengestellt und bieten diese als Komplettpaket mit erweiterten Funktionen und Anwender-Software an. Diese Zusammenstellungen werden Distributionen genannt. Je nach Verwendung wird Linux an verschiedene Bedürfnisse angepasst,

[29] *Glatz, Eduard: Betriebssysteme. Grundlagen, Konzepte, Systemprogrammierung, S. 599, Heidelberg 2006.* [28]

beispielweise bei speziellen Hardwarekomponenten. Die meisten Distributionen bieten eine oder mehrere grafische Oberflächen an und bringen eine Vielzahl von Programmen mit. Aufgrund der GNU-Lizenz müssen diese meistens kostenlos zur Verfügung gestellt werden. Sie finanzieren sich durch Spenden. Andere schaffen es, sich durch Support und Spezialfunktionen, die sie in ihre Distribution einbauen, zu finanzieren. An dieser Stelle sind *Red Hat* und *Oracle-Linux* als Beispiele zu nennen. Weitere bekannte Distributionen sind *Android*, *Ubuntu* und *Suse*. [30]

2.4.2 Logsystem in Linux

Als Betriebssystem verfügt Linux über ein Nachrichtensystem, was wichtige Meldungen und Informationen der Anwendungsprogramme, Dienste (Daemons) und des Systems aufzeichnet. Diese Aufzeichnungen werden im Allgmeinen als „Logs“ bezeichnet. Die Datei welche die Nachrichten enthält nennt sich „Log-Datei“. Für die Verifikation von Stabilität und zur Prüfung des ordnungsgemäßen Betriebs eines Linuxsystems ist die Analyse dieser Benachrichtigungen hilfreich. In *Oracle-Linux* sowie in den meisten anderen Linux-Distributionen, werden die Log-Dateien in dem Verzeichnis `/var/log/` gespeichert. In jedem System variiert der Inhalt des Ordners, je nachdem welche Programme installiert sind. Im Wesentlichen befinden sich dort einzelne Dateien, in denen die auftretenden Nachrichten chronologisch abgespeichert werden. Verschiedene Programme haben ihre eigene Log-Datei, beispielsweise *Yum* und der *Automatic Storage Management* (ASM) von *Oracle*. Umfangreiche Programme, die mehrere Log-Dateien erstellen, speichern diese in einem separaten Unterordner. Hier sind *Samba* und *CUPS* (Common Unix Printing System) als Beispiele zu nennen.

Umgekehrt existieren auch Dateien, welche die Nachrichten von mehreren kleinen Programmen beinhalten. Der Syslog-Dienst nimmt solche Benachrichtigungen entgegen und schreibt sie gemeinsam in eine Datei. Beispiele dafür sind `daemon` und `user`. In Übersicht 2.1 sind Beispiele für Log-Dateien und ihre Inhalte aufgelistet. [31]

In der Datei `/etc/rsyslog.conf`, in manchen Betriebssystemen auch nur `syslog.conf`, kann das Log-System konfiguriert werden. Dafür werden Benachrichtigungsregeln definiert. Jede dieser Regeln besteht aus einem Paar: Nachrichtenquelle und -ziel. Letzteres kann eine Log-Datei, eine Konsolensitzung eines Nutzers oder ein Syslog-Dienst im Netzwerk sein. Die Quelle besteht wiederum aus einer „Facility“ (Herkunftskategorien), die durch einen Punkt mit der „Priority“ (Priorität) verknüpft ist (`Facility.Priority`). Das folgende Beispiel verdeutlicht den Aufbau:

```
user.warn /var/log/user.log
```

[30] *Vgl. Plötner, Johannes / Wendzel, Steffen: Einstieg in Linux. Linux verstehen und einsetzen, 5. Auflage, S. 21f, Bonn 2012.* [29]

[31] *Vgl. Wikibooks (Hrsg.): Linux-Praxisbuch: Syslog, 2015.* [30]

Beschreibung	Typ
dmesg	Enthält Nachrichten vom Kern. Die Informationen werden zyklisch überschrieben (Ringpuffer). Die Datei dmesg.old enthält auch ältere Meldungen.
messages	Allgemeine Informationen zum System
syslog	Nachrichten zum Syslog-Dienst selber
oracleasm	Log-Datei des ASM-Grid-Systems
maillog	Log des Mailservers (nur wenn installiert)
yum.log	Fehler und Probleme im Paketmanager *Yum*
samba	Log-Datei von *Samba*
cron	Enthält Nachrichten von Cron
boot.log	Fehler und Probleme beim Bootvorgang
secure	Logt Authentifikationsfehler, wie z.B. eine falsche Anmeldung

Tabelle 2.1: Beispieldateien aus dem Ordner /usr/log/ [31]

Die Facility „user“, die Nachrichten von Nutzerprozessen zurückgibt, wird mit der Priority „Warnung“ verknüpft. Alle Nachrichten von Nutzerprozessen, die eine höhere Priorität als „Warnung“ haben, werden in die Datei „user.log“ geschrieben. Mehrere Paare an Facility und Priority können durch ein Semikolon getrennt aneinandergereiht werden. [32] In den Tabellen F.2 und F.3, aus Anlage F.2, sind mögliche Facilities und Priorities aufgelistet.

2.4.3 Pakete und Paketmanager

Ein Paket, im Sinne von Linux, ist eine Datei, welche die zu installierenden Daten enthält. Neben der Nutzlast des Paketes, also der Anwendung, Bibliotheken oder anderer Daten, beinhaltet das Paket Informationen, die den Inhalt näher beschreiben. Die Versionsnummer, der Paketersteller oder die Anforderung die ein Paket an das System stellt, sind solche Informationen. Diese werden im Header der Datei gespeichert. Verschiedene Programme benötigen wiederum andere Programme oder Bibliotheken, um richtig zu funktionieren. Da diese auf unterschiedliche Pakete verteilt sind, entstehen so Abhängigkeiten zwischen ihnen. Es existieren verschiedene Paket-Formate, am meisten verbreitet sind *RPM* und *DEB* (Debian-Pakete) .

Pakete werden in der Regel auf einem zentralen Server gelagert und mithilfe eines Paketmanagers von diesem heruntergeladen. Dies ist ein Programm, mit dem Pakete installiert, deinstalliert, aktualisiert und andere Verwaltungsoperationen durchgeführt werden. Der Paketmanager protokolliert alle Vorgänge, um immer Auskunft über die installierte Software leisten zu können. Moderne Paketmanager können die Abhängigkeiten zwischen Paketen erkennen und auflösen. So werden alle Pakete automatisch mit installiert, die das System benötigt, um eine Anwendung zu betreiben. Eine wichtige

[32] *Vgl. Oracle Corporation (Hrsg.): Oracle Linux Security Guide for Release 6, Configuring and Using System Logging, 2015.* [32]

Funktion ist das Auswerten von Signaturen. Ein Paket mit einer falschen Signatur könnte bei der Übertragung beschädigt oder von einem Angreifer durch ein infiziertes Paket ausgetauscht worden sein. Mit einem signierten Paket wird eine fehlerfreie Übertragung vom Repository sichergestellt. [33]

2.5 Vorstellung relevanter Oracle Software

Die Firma *Oracle* bietet eine Vielzahl an Softwareprodukten an, der Fokus liegt dabei auf professionellen Serveranwendungen. Dazu zählen Virtualisierungslösungen, Middleware, Managementsoftware, die Programmiersprache *Java*, Datenbanken und Betriebssysteme. Für diese Arbeit ist die auf *Red Hat-Linux* basierende Distribution *Oracle-Linux*, und die *Oracle-Datenbank* relevant. Die folgenden zwei Abschnitte geben einen Einblick in diese Software.

2.5.1 Oracle-Linux

Oracle-Linux ist ein für den Serverbereich ausgelegtes Betriebssystem und steht unter der GNU-GPL (GNU-General Public License). Daher ist die Nutzung von *Oracle-Linux* kostenlos. Der Support durch *Oracle* kostet jedoch Geld. Ein einjähriger Zugang zum *Oracle-Linux-Netzwerk* kostet 119$, eine dreijährige Mitgliedschaft als *Oracle Linux Premier* kostet bereits 6.897$. [34] *Oracle* hält seine Distribution kompatibel zu den Paketen aus dem *Red Hat*-Repositories und bietet einen *Red Hat* kompatiblen Kern an. So wird die Stabilität und Kompatibilität der Programme von *Red Hat* mit in *Oracle-Linux* übernommen. Programme, die lange und ausgiebig auf Sicherheit und Stabilität unter *Red Hat* getestet wurden, können so auch unter *Oracle-Linux* betrieben werden. Zugleich bietet *Oracle* seine Datenbanksysteme für *Oracle-Linux* an und erweitert die Distribution um einige Funktionen.

Der *Unbreakable Enterprise Kernel* (UEK) ist ein von *Oracle* angepasster Linux-Kern, welcher insbesondere auf Stabilität für Serverhardware und für *Oracle*-Datenbanken optimiert ist. Die neuesten Funktionen von *Oracle* werden zuerst in diesen Kern eingepflegt, so war das Live-Patching zuerst nur für den UEK verfügbar. Das Gegenstück zum *Unbreakable Enterprise Kernel* ist der *Red Hat Compatible Kernel* (RCK). Das System kann mit beiden Kernen betrieben werden, für den Wechsel ist ein Neustart nötig.

Weiterhin werden mehrere Dateisysteme unterstützt, die vor allem für den Serverbereich interessant sind, beispielsweise *XFS* (X-File System), *Btrfs* (B-tree File System) oder das *Oracle-Cluster-Dateisystem*. Die Analyse-Software *DTrace*, welche ursprünglich von

[33] *Vgl. Siever, Ellen u.a.: LINUX in a Nutshell, 6. Auflage, S. 542-595, Beijing u.a. 2009.* [34]

[34] *Vgl. Oracle Corporation (Hrsg.): Oracle Linux Support and Oracle VM Support Global Price List, S. 2f, 2014.* [35]

Sun Microsystems entwickelt wurde, steht auch unter *Oracle-Linux* zur Verfügung. [35]

Als *Unbreakable Linux Network* (ULN) benennt *Oracle* den Zugang zu verschiedenen Instanzen seines Supports. Neben Informationswebseiten, Dokumenten und Foren zählen auch Repository-Server zu den Bestandteilen des ULN. Über diese beziehen *Oracle*-Kunden Updates für ihre Produkte und es stehen zusätzliche Programme zur Verfügung. [36]

2.5.2 Oracle-Datenbank und ASM Grid

Die Firma *Oracle* bietet verschiedene Datenbanklösungen an, darunter zählen *MySQL* und die *Oracle*-Datenbank. Die auch für den Serverbereich ausgelegten Datenbanken werden von einer Vielzahl weiterer Software unterstützt, um den Anforderungen auf einem Serversystem zu entsprechen.

In dieser Arbeit wird auf Software, welche für die *Oracle*-Datenbank zur Verfügung stehen, Bezug genommen. Diese sind die *Oracle Grid Infrastruktur*, *Real Application Clusters* (RAC) und *Automatic Storage Management* (ASM).

Der Speicherplatzbedarf einer Datenbank wächst mit der Zeit und Speichermedien müssen aus verschiedensten Gründen getauscht werden. Daher ist es sinnvoll die Verwaltung des Festplattenspeichers möglichst skalierbar zu gestalten. Für diese Aufgabe gibt es „Volumen-Manager“. Diese Gruppe von Software ermöglicht es, über mehrere Speichermedien ein logisches Volumen zu definieren und dieses nach belieben aufzuteilen oder zu erweitern. Der *Oracle*-Vertreter dieser Gruppe heißt *Automatic Storage Management*. [37]

Nicht nur der Festplattenspeicher, sondern auch andere Ressourcen, wie Prozessorleistung und Arbeitsspeicher, müssen aufgrund unterschiedlicher Faktoren, wie Auslastungsschwankungen, Wachstum, Modernisierung und Hardwarefehler, verwaltet werden. Dabei hilft ein Verbund mehrerer Server, die gemeinsam eine größere Aufgabe bewältigen. Dieser Zusammenschluss von Computern wird in der Informatik als „Grid“ oder „Cluster“ bezeichnet. Die Firma *Oracle* benennt seinen Vertreter *Real Application Clusters*. Diese Anwendungen werden zusammen mit einer Datenbank auch als „Grid Infrastruktur“ bezeichnet. Unter diesen Begriff fallen auch andere Programme, die für diese Arbeit aber nicht relevant sind. [38] Dieser Zusammenhang wird benötigt, um die

[35] *Vgl. Oracle Corporation (Hrsg.): Oracle Data Sheet, Oracle Linux, 2014.* [36]

[36] *Vgl. Oracle Corporation (Hrsg.): Oracle Linux Security Guide for Release 6, About the Unbreakable Linux Network, 2015.* [37]

[37] *Vgl. Hunter, Jeffrey: DBA Tips Archive for Oracle, Install Oracle Database 11g R2 on Linux using Oracle ASM - (OL5), 2015.* [38]

[38] *Vgl. Solbach, Sebastian: Applikationsüberwachung mit 11gR2 Grid Infrastruktur - Am Beispiel der DBConsole.* [39]

spätere Prozessstruktur der *Oracle*-Datenbank zu verstehen. Denn die Datenbank wird im Sinne eines Grid-Setups nur als eine Ressource des Clusters angesehen. Bevor man die Datenbank also starten und deren Prozesse analysieren kann, müssen die entsprechenden Grid-Systeme und ASM gestartet sein. Die Prozessarchitektur und die Zusammenhänge aller beteiligten Prozesse zu beleuchten wäre sehr umfangreich und ist für diese Arbeit nicht nötig. Im Folgenden soll nur eine kleine Übersicht der *Oracle* Datenbank-Prozesse gegeben werden.

Die *Oracle*-Datenbank besteht nicht aus einem einzelnen Prozess, sondern aus mehreren. Diese lassen sich in Datenbankprozesse und Nutzerprozesse unterteilen. Dazu kommen noch Prozesse des ASM Systems, wenn es denn eingerichtet ist. Die Nutzerprozesse werden für jeden an der Datenbank angemeldeten Nutzer erzeugt. Diese verwalten die Kommunikation von der Datenbank nach außen und umgekehrt. Ob ein Nutzerprozess einen eigenen Serverprozess zugeordnet bekommt oder sich diesen mit anderen Prozessen teilen muss, hängt von der Serverkonfiguration ab. Datenbankprozesse können in Hintergrundprozesse und Serverprozesse unterteilt werden. Letztere verarbeiten die Anforderungen der Nutzerprozesse. In einigen Konfigurationen werden diese Prozesse zusammengeführt, um die Leistung der Datenbank zu erhöhen. Die Hintergrundprozesse einer *Oracle*-Datenbank haben unterschiedliche Aufgaben und werden nur bei Bedarf gestartet oder beendet. Zu ihren Funktionen gehören unter anderem das Schreiben von Log-Dateien, die Überwachung der Datenbank und das zyklische Sichern von Daten-Puffern. [39] [40]

2.6 Stand der Technik

In der sich stetig und schnell weiterentwickelnden IT-Welt, ist es wichtig Aussagen in einen zeitlichen Rahmen einzuordnen. Die folgenden Abschnitte geben einen Einblick in den technischen „Ist-Zustand“, zum Erstellungszeitpunkt der Arbeit. Das ermöglicht dem Leser die Arbeit besser in einen technischen Kontext einzuordnen.

2.6.1 Dynamische Software Updates

Als *Dynamische Software Updates* (DSU) werden die Updates bezeichnet, welche sich während der Laufzeit eines Programms einspielen lassen. DSU wird auch Live-Patching genannt. Im Jahr 1983 schrieb bereits *Insup Lee* in seiner Dissertation „DYMOS: A DYNAMIC MODIFICATION SYSTEM“ über das Thema. Mit seiner dynamischen Updatesoftware *DYMOS* war ein Programmierer in der Lage ein modulares Programm während

[39] *Vgl. Oracle corp. (Hrsg.): Oracle Help Center - Database Concepts, Process Architecture, S. 23, 2015.* [40]

[40] *Vgl. Oracle corp. (Hrsg.): Oracle Help Center - Database Administrator's Guide, About Oracle Database Background Processes, S. 37, 2015.* [41]

der Laufzeit zu updaten. [41] Bis zum jetztigen Zeitpunkt existiert nur eine überschaubare Anzahl an DSU-Systemen. Die folgende Übersicht gibt einen Einblick:

- *Ginseng* ist eine DSU-System für Linux, es wurde unter anderem für die Aktualisierung von *Vsftpd*, *OpenSSH* und *GNU-Zebra* genutzt. [42]
- *Kitsume* ist eine C-Framework, welches die Entwicklnug von DSU fähiger Software unterstützt. [43]
- *UpStare* ist ein DSU-System zum Updaten von C-Programmen. [44]
- *DynAMOS* ist ein Live-Patching-System für den Betriebssystemkern. [45]
- *PoLUS* ist ein weiteres DSU-System [46]
- *Erlang* ist eine Programmiersprache und Laufzeitumgebung, welche das Auswechseln von Modulen zur Laufzeit unterstützt. [47]
- *Kgraft* und *Kpatch* sind zwei Programme zum „live patchen“ eines Linux-Kerns. Sie wurden von *Suse* (Kgraft) und *Red Hat* (Kpatch) entwickelt. [48]

2.6.2 Der „aktuelle“ Linux-Kern

Am 30. August 2015 erschien der Linux-Kern in der Version 4.2. [49] Er enthält Funktionen für die Verschlüsselung von Dateisystemen, Grafikkartentreiber, Live-Patching und Mechanismen für automatische UEFI Updates.

Die interessanteste Erneuerung für diese Arbeit ist das Live-Patching des Kerns. Es wurde in die Kern-Version 4.0 aufgenommen und in den Nachfolgerversionen weiterentwickelt. Es ermöglicht wie *Ksplice* das Aktualisieren des Kerns während seiner Laufzeit. Diese Funktion basiert auf den beiden *Ksplice*-Ablegern *Kpatch* und *Kgraft*, die von *Red Hat* und *Suse* entwickelt wurden. [50] Linux ist zur Zeit das einzige Betriebssystem, welches sich während der Laufzeit aktualisieren kann.

2.6.3 Aktuelle Paketmanager

Ein Paketmanager ist ein sehr nützliches Programm und seit über zehn Jahren Standard in der Linux-Welt. Sie können danach unterteilt werden, welche Paketarten sie in den unterschiedlichen Distributionen unterstützen. Bei dieser Unterteilung entstehen zwei

[41] *Vgl. Lee, Insup: DYMOS: A DYNAMIC MODIFICATION SYSTEM, 1983.* [42]
[42] *Vgl. University of Maryland (Hrsg.): DYNAMIC SOFTWARE UPDATING, 2011.* [43]
[43] *Vgl. Smith, Edward u.a.: KITSUNE, 2013.* [44]
[44] *Vgl. Makris, Kristis: UpStare manual, Release 0-12-9, S.4, 2012.* [45]
[45] *Vgl. The DynAMOS Team (Hrsg.): DynAMOS manual Release 0-5-5, S.1, 2007.* [46]
[46] *Vgl. Chen, Haibo u.a.: POLUS: A Powerful live updating system, 2007.* [47]
[47] *Vgl. Ericsson Computer Science Laboratory (Hrsg.): ERLANG, GETTING STARTED.* [48]
[48] *Vgl. Thoma, Jörg / Kißling, Kristian: Linux-Kernel 4.0 bringt Live-Patching, 2015.* [49]
[49] *Vgl. The Linux Kernel Organization, Inc. (Hrsg.): The Linux Kernel Archives, San Francisco 2015.* [50]
[50] *Vgl. Thoma, Jörg / Kißling, Kristian: Linux-Kernel 4.0 bringt Live-Patching, 2015.* [49]

Gruppen, die Paketmanager welche *DEB*-Pakete und jene die *RPM*-Pakete verwalten. In der Distribution *Oracle*-Linux wird *Yum* (*RPM*) als Paketmanager eingesetzt. Dessen neueste Version ist am 28. Juni 2011 (4.3.4) erschienen. [51] Unter dem Namen *DNF* (Dandified Yum) wurde *Yum* seit 2012, in einem neuen Projekt, weiterentwickelt (Fork). Mittlerweile ist *DNF* der Standardpaketmanager in der Distribution *Fedora* und hat damit *Yum* abgelöst. [52] Ob Serverdistributionen wie *Red Hat*, *CentOS* oder *Oracle-Linux* auch *Yum* durch *DNF* ersetzen, konnte nicht recherchiert werden. Auch die Firma *Microsoft* hat in seinem neuesten Betriebssystem *Windows 10* einen Paketmanager namens *OneGet* eingebaut. [53] Bei Betriebssystemen für mobile Endgeräte, wie Smartphone oder Tablet, sind so genannte *App-Stores* integriert, die ähnliche Funktionen wie ein Paketmanager bieten. Darin lässt sich ein Trend zu zentral verwalteten Quellen für Anwendungsprogramme erkennen.

[51] *Vgl. Open Source Lab (Hrsg.): Yum Package Manager, 2014.* [51]
[52] *Vgl. Fischer, Marcus: Der „neue“ Paketmanager DNF, 2015.* [52]
[53] *Vgl. Heilig, Tobias: OneGet – der neue Paketmanager unter Windows 10, 2015.* [53]

3 Ksplice das Oracle-Linux Live-Patching-System

Dieses Kapitel fokussiert sich auf die Software *Ksplice*, die es ermöglicht Updates für den Linux-Kerns während der Laufzeit einzuspielen. Sie wurde ursprünglich von dem gleichnamigen Unternehmen *Ksplice Inc*. entwickelt, bis dieses im Juli 2011 von *Oracle Corp.* erworben wurde. [54] Die ersten Teilabschnitte des Kapitels stellen die Software und ihre Funktion vor, während im zweiten Abschnitt der technische Hintergrund und die Funktionsweise von *Ksplice* erläutert werden.

3.1 Vorstellung von Ksplice

Ksplice wird von *Oracle* nur für Kunden mit *Oracle Linux Premier Support* zur Verfügung gestellt. Diese erhalten so eine Möglichkeit, die regelmäßigen Sicherheitsupdates für *Oracle-Linux* einzuspielen, ohne das Betriebssystem neu starten zu müssen. [55] Zur Zeit unterstützt *Ksplice* Linux-Kerne von zehn unterschiedlichen Distributionen, wobei die Rahmenbedingungen für verschiedene Kunden und Distributionen unterschiedlich geregelt sind. [56] *Oracle* stellt *Ksplice* kostenlos für die Linuxdistributionen *Ubuntu* und *Fedora* zur Verfügung, bietet aber keinen Support und keine Garantie. [57]

3.2 Funktionsweise

Das Kommandozeilentool von *Ksplice*, welches für die Verwaltung der Patches verantwortlich ist, nennt sich *Uptrack*. Um *Ksplice*-Patches zu nutzen, werden ein Zugangsschlüssel und die Software *Uptrack* auf dem Betriebssystem benötigt, welches *Ksplice*-Patches beziehen soll. *Uptrack* wird auf unterschiedliche Weise installiert, je nachdem für welche Distribution es gebraucht wird und ob ein Zugang zum *Unbreakable Linux Network* zur Verfügung steht. Für die Distributionen *Ubuntu* und *Fedora* wird keine Registrierung benötigt, da für diese eine Installationsdatei als *DEB*- oder *RPM*-Paket heruntergeladen werden kann. [57] *Oracle*-Kunden müssen einen Zugangsschlüssel über das ULN beziehen und können *Uptrack* durch ein Script installieren. [58] Eine einfachere Möglichkeit

[54] *Vgl. Oracle Corporation (Hrsg.): Oracle Buys Ksplice. Oracle Linux Enhanced with Zero Downtime Software Updates, Redwood Shores Calif. 2011.* [54]
[55] *Vgl. Coekaerts, Wim / Oracle Corporation (Hrsg.): Using Oracle Ksplice to Update Oracle Linux Systems Without Rebooting, Revision 1.0, 2011.* [55]
[56] *Vgl. Oracle Corporation (Hrsg.): Customer Support.* [56]
[57] *Vgl. Oracle Corporation (Hrsg.): Ksplice Desktop.* [57]
[58] *Vgl. Oracle Corporation (Hrsg.): Uptrack Installing Uptrack.* [58]

ist es, sein System im ULN zu registrieren und *Uptrack* über den Paketmanager zu installieren. Dabei wird der Zugangsschlüssel von *Uptrack* selber aus dem ULN heruntergeladen. [59] Wenn keine Verbindung zum Internet besteht, kann ein selbst administrierter ULN-Mirrow-Server im Netzwerk genutzt werden. Dieses Setup wird näher in Abschnitt 3.3.4 auf Seite 24 beschrieben. *Uptrack* versorgt immer nur das System mit Updates, auf dem es installiert ist. Nicht jeder Linux-Kern wird unterstützt, das sollte vor einer Installation geprüft werden. Für *Oracle*-Linux trifft das für folgende Kerne zu [60]:

- Alle Oracle *Unbreakable Enterprise Kernel* für *Oracle-Linux* 7, beginnend mit Version 3.8.13-35 (erschienen am 13.05.2014).
- Alle Oracle *Unbreakable Enterprise Kernel* für *Oracle-Linux* 5 und 6, beginnend mit Version 2.6.32-100.28.9 (erschienen am 16.03.2011)
- Alle *Oracle-Linux* 7 Kerne, ab der offiziellen Veröffentlichung.
- Alle *Oracle-Linux* 6 Kerne, ab der offiziellen Veröffentlichung.
- Alle *Oracle-Linux* 5 *Red Hat Compatible Kernels*, ab der *Oracle-Linux*-Version 5.4 (2.6.18-164.el5, erschienen am 09.08.2009).
- Alle *Oracle-Linux* 5 *Red Hat Compatible Kernels*, mit integrierten Patches, ab *Oracle-Linux*-Version 5.6 (2.6.18-238.0.0.0.1.el5, erschienen am 22.01.2011).

3.3 Bestandteile

Ksplice kann neben seinen Kernfunktionen mit einigen weiteren Eigenschaften aufwarten. So ist es möglich, eingespielte Updates mit einem „Rollback“ rückgängig zu machen oder die Updates über einen eigenen Update-Server herunterzuladen. Letzteres ist besonders interessant für Unternehmen, die ihre Server nicht direkt mit dem Internet verbunden haben. *Ksplice* bietet ein Webinterface an, mit dem die Aktualität der Systeme überwacht werden kann. Da *Ksplice* kein permanent laufendes Programm oder Dienst ist, kommt es nicht zu einem unnötigem Verbrauch von Systemressourcen während des Standardbetriebs. Die Installation der Software selber benötigt keinen Neustart. Ebenso gehören E-Mail-Benachrichtigungen, signierte Updates und der Einsatz mit virtuellen Maschinen zum Funktions-Portfolio von *Ksplice*. [61]

3.3.1 Ksplice-Uptrack

Ksplice-Uptrack ist das Kommandozeilentool von *Ksplice*, das zum Verwalten der *Ksplice*-Patches genutzt wird. Im Folgenden wird *Ksplice-Uptrack* nur noch als *Uptrack* bezeichnet. Mit Hilfe dieser Software können die Updates installiert und deinstalliert

[59] *Vgl. Oracle Corporation (Hrsg.): Oracle Linux Ksplice User's Guide, Registering to Use Oracle Ksplice, 2015.* [59]
[60] *Vgl. Oracle Corporation (Hrsg.): Customer Support.* [56]
[61] *Vgl. Oracle Corporation (Hrsg.): Features.* [60]

werden. Zudem können die verfügbaren Updates angezeigt sowie Einstellungen vorgenommen werden. Nur für die Distributionen *Fedora* und *Ubuntu* gibt es momentan eine grafische Nutzerschnittstelle. Bei der Verwaltung der Updates auf mehreren Systemen wird der Administrator durch ein Webinterface unterstützt. Über eine Richtlinie kann festlegt werden, ob der Server selbstständig Updates herunterladen und installieren darf oder nicht [66].

3.3.2 API-Tools

Uptrack bietet Entwicklern eine Schnittstelle zur Anbindung eigener Software. Diese wird in zwei Formen bereit gestellt. Einmal als REST-API (Representational State Transfer - Application Programming Interface) und einmal in Form einer Python-API. Mithilfe von REST können über einen Webdienst Ressourcen bzw. Informationen ausgeliefert oder verändert werden. Dafür werden übliche Internettechnologien, wie HTTP (Hypertext Transfer Protocol) und HTML (Hypertext Markup Language), verwendet. Lediglich eine URI (Uniform Resource Identifier), die an den REST-Dienst übermittelt wird, reicht aus, um Informationen zu lesen, zu löschen oder zu verändern.

Über die *Uptrack*-API können Informationen zu den einzelnen Kernen ausgelesen und verändert werden. Darunter fallen die Gruppenzugehörigkeit, die Anzahl der installierten oder offenen Patches und Berechtigungen zum Einspielen von Patches. [62] Informationen über die Systeme, auf denen *Uptrack* betrieben wird, können auf diese Weise in ein eigenes Nutzerinterface eingearbeitet werden. Die API funktioniert weder für *Uptrack*-Clients, die nicht bei *Ksplice* registriert sind, noch für Offline-Clients. [63]

3.3.3 Ksplice-Webinterface

Das Webinterface von *Uptrack* ist nur für Abonnenten bzw. Kunden von *Oracle* nutzbar, welche ihr System im ULN registriert haben. Es listet in einer Übersicht alle verwalteten Linux-Kerne auf und stellt nützliche Informationen dar, beispielsweise welche Version des Linux-Kerns gerade verwendet und welche Patches installiert sind. momentan installiert ist. Das Webinterface benachrichtigt seinen Nutzer über neue *Ksplice*-Patches, Updates für *Uptrack* und nicht mehr mit dem ULN in Verbindung stehende Systeme. Für jeden *Uptrack*-Client gibt es eine Detailansicht, welche Informationen zu den Updates und zum System darstellt. Zudem ist es möglich, das Einspielen von *Ksplice*-Patches für einzelne Systeme zu deaktivieren. [64]

---

[62] *Vgl. Oracle Corporation (Hrsg.): Ksplice Uptrack API.* [61]
[63] *Vgl. Oracle Corporation (Hrsg.): Ksplice User's Guide 2.1 About the Ksplice Uptrack API, 2015.* [62]
[64] *Vgl. Oracle Corporation (Hrsg.): Uptrack Uptrack Web Interface.* [63]

3.3.4 Uptrack offline Verwendung

Zur Aktualisierung mittels *Ksplice* bietet *Oracle* zwei Methoden an. Bei der Standardmethode werden die *Ksplice*-Patches über das Internet aus dem *Unbreakable Linux Network* von *Oracle* geladen. Eine andere Möglichkeit ist es, einen lokalen Spiegelserver (ULN-Mirrow) des *Oracle*-ULN im eigenen Netzwerk aufzusetzen. Von diesem können die Computersysteme, die sich im gleichem Netzwerk befinden, die *Ksplice*-Patches beziehen. So können die Linux-Kerne mit *Ksplice*-Patches versorgt werden, auch wenn sie nicht direkt mit dem Internet verbunden sind. Der ULN-Mirrow bietet damit eine höhere Sicherheit, sowie eine bessere Administration der zur Verfügung gestellten Updates. [65]

3.4 Der Updatevorgang

Systeme, die *Ksplice*, einsetzen haben zwei Möglichkeiten ihren Kern zu aktualisieren. Bei der klassischen Variante wird ein Update mit dem Paketmanager von einem Repository-Server heruntergeladen und installiert. Die neue Version des Linux-Kerns kann dabei erst genutzt werden, wenn das System neu gestartet wurde. Im Folgenden werden solche Linux-Kerne als „Original-Kern“ bezeichnet. Bei der zweiten Möglichkeit wird *Uptrack* genutzt. Dieses lädt, auch von einem Repository-Server, einen *Ksplice*-Patch herunter und installiert ihn. Die Besonderheit bei diesen Patches ist es, dass sie direkt in den Arbeitsspeicher geladen werden und damit das System nicht neu gestartet werden muss. Die auf diese Weise eingespielten Versionen des Linux-Kernes werden im Folgenden als „Effektiv-Kern“ bezeichnet. Vom Administrator kann gewählt werden, ob diese Updates automatisch eingespielt werden sollen oder nicht.

Bei einem Neustart des Betriebssystems wird zuerst der „Original-Kern“ geladen und anschließend die *Ksplice*-Patches wieder eingespielt, so dass der gleiche „Effektiv-Kern“ genutzt wird wie vor dem Neustart. Dies geschieht während des Bootvorgangs, so dass nach dem Start des Systems alle Patches wieder im Kern installiert sind. Dadurch soll die Sicherheit und die Stabilität bestehen bleiben.In der Konfigurationsdatei von *Uptrack*, welche im Verzeichnis `/etc/uptrack/uptrack.conf` zu finden ist, kann das automatische Einspielen der Patches beim Bootvorgang abgeschaltet werden. Hier können auch noch weitere Einstellungen vorgenommen werden, wie das automatische Installieren von Patches oder E-Mail-Benachrichtigungen.

Parallel zu den *Ksplice*-Patches kann der neue „Original-Kern“ per Paketmanager installiert werden. Wenn der Server aus anderen Gründen, beispielsweise eines Hardwareupdates, neu gestartet werden muss, kann direkt in den neuen „Original-Kern“ gebootet werden.

[65] *Vgl. Oracle Corporation (Hrsg.): Ksplice Offline Client, Overview.* [64]

Unvorteilhaft ist dieses Vorgehen, wenn bestimmte Programme oder Treiber speziell für einen „Original-Kern“ kompiliert wurden. Hier ist es sinnvoll, den „Original-Kern“ beizubehalten und nicht über den Paketmanager zu aktualisieren. [66]

Ksplice aktualisiert nur den Linux-Kern und keine anderen Teile des Betriebssystems. Die meisten normalen Patches des Linux-Kerns lassen sich in *Ksplice*-Patches umwandeln, die wenigsten müssen durch einen Programmierer angepasst werden. Dies ist der Fall, wenn der Patch semantische Änderungen an Datenstrukturen des Kerns vornimmt. Beispiele dafür sind das Initialisieren von Datenstrukturen, das Hinzufügen von Feldern oder die Änderung einer Variablendeklaration. *Ksplice*-Patches sind sehr klein, in den meisten von ihnen werden durchschnittlich weniger als 15 Zeilen Quellcode verändert. [67] Eine Beispielliste von *Ksplice*-Patches befindet sich im Anhang F.3 auf Seite 125.

3.5 Funktionsweise der Ksplice-Patches

Dieser Abschnitt erläutert, wie *Ksplice* den Linux-Kern während der Laufzeit aktualisiert und was dabei beachtet werden muss. Die Abbildung 3.1 verdeutlicht dabei den Ablauf.

3.5.1 Theoretischer Ablauf

Um ein neues Update in das laufende System einzuspielen, ermittelt *Ksplice* die veränderten und hinzugefügten Funktionen. Dabei wird nicht der Quellcode, sondern der Binärcode verglichen. Um diese zu erhalten, wird der Kern zwei Mal kompiliert, einmal ohne Aktualisierungen und einmal mit den geladenen *Ksplice*-Patches. Die Binärdaten, die ohne das Update erstellt wurden, werden als „Pre-Code“ bezeichnet, die mit Updatefunktionen als „Post-Code“. Nach dem Kompilieren wird der Post- und Pre-Code miteinander verglichen. Dabei wird aus jeder Veränderung ein einzelner Veränderungsabschnitt erstellt, ein so genannter „Splice“. Für die Funktionen, die zu dem Kern neu hinzukommen, werden auch Splices erstellt. Diese werden alle in einer eigenen Datei zusammengefasst, woraus ein Kernelmodul entsteht. [67]

Aus der Datei mit den gebündelten Splices wird ein Kernelmodul erstellt, welches die neuen Funktionen in den Kern laden soll. Dieses Modul kann aber noch nicht verwendet werden, da sich die neuen Funktionen noch nicht auf die richtigen Adressen im realen Speicher beziehen. *Ksplice* vergleicht nun byteweise den Pre-Code mit dem im Arbeitsspeicher liegenden Run-Code. Dieser Pre-Run-Vergleich dient zum Ermitteln der noch benötigten Adressen.

---

[66] *Vgl. Oracle Corporation (Hrsg.): Uptrack User's Guide.* [65]

[67] *Vgl. Arnold, Jeff / Kaashoek Frans: Ksplice: Automatic Rebootless Kern Updates, S.2-8, Nuremberg 2009.* [66]

Wenn dabei eine Unstimmigkeit aufgrund eines zweideutigen Bezeichners auftritt, berechnet *Ksplice* die benötigte Speicheradresse anhand der Informationen aus dem Run-Code. [68]

Nachdem alle zu ersetzenden Splices und die entsprechenden Adressen ermittelt wurden, wird das Kernelmodul eingespielt. Dabei werden alle normalen Operationen des Systems mithilfe der Funktion `stop_machine` unterbrochen. Diese Unterbrechung dauert ca. 0.7 Millisekunden, was für die meisten Systeme unbedeutend ist. Während dieser Unterbrechung verlinkt *Ksplice* auf die neuen Funktionen, die in den Speicher geladen wurden. [68]

Es können jedoch nur die Funktionen ersetzt werden, die nicht gerade verwendet werden. Das heißt, dass jede Funktion auf ihre Aktivität geprüft wird. Alle nicht aktiven Funktionen können ersetzt werden. Wenn jedoch ein Thread-Pointer oder ein Eintrag aus dem Kernelstack direkt auf eine Adresse in der Funktion verweist, dann wird diese als aktiv betrachtet und nicht ersetzt. Wenn es nicht gelingt, diese Funktion zu ersetzen, muss *Ksplice* das Update abbrechen. [69]

3.5.2 Objekt-Schicht

Ksplice arbeitet auf der Objekt-Schicht, d.h., es tauscht den Code nur funktionsweise und nicht jede Codezeile einzeln. Wenn sich in einer Funktion eine Variable ändert, wird die ganze Funktion ausgetauscht. Dies hat verschiedene Vorteile, so muss man nicht die Feinheiten in der Programmiersprache beachten, z.B. das „Casting“ von Datentypen und die Geltungsbereiche von Variablen. Denn es ist schwerer, diese kleinen Änderungen im Binärcode zu identifizieren und auszutauschen als die ganze Funktion zu ersetzen. Auch die Ermittelung der Speicheradresse wird dadurch beschleunigt. Denn es müssen nur die Bezeichner der Funktionen in der Symboltabelle gesucht werden und nicht jede Variable. [68]

Mit dem funktionsweisen Austausch kann noch ein weiteres Problem umgangen werden. Der Binärcode ist umfangreich und enthält relative Sprungebefehle (Jump-Instructions) zu anderen Abschnitten seines Codes. Wenn versucht wird, eine einzelne Zeile neuen Codes einzufügen, dann würden die Adressen der relativen Sprünge nicht mehr auf die korrekten Ziele zeigen. Das Programm wäre somit unbrauchbar. [68]

---

[68] *Vgl. Arnold, Jeff / Kaashoek Frans: Ksplice: Automatic Rebootless Kern Updates, S. 2-8, Nuremberg 2009.* [66]

[69] *Vgl. Streicher, Martin: Speaking UNIX: Get to know Ksplice. Give reboots the boot, S 3, 2010.* [67]

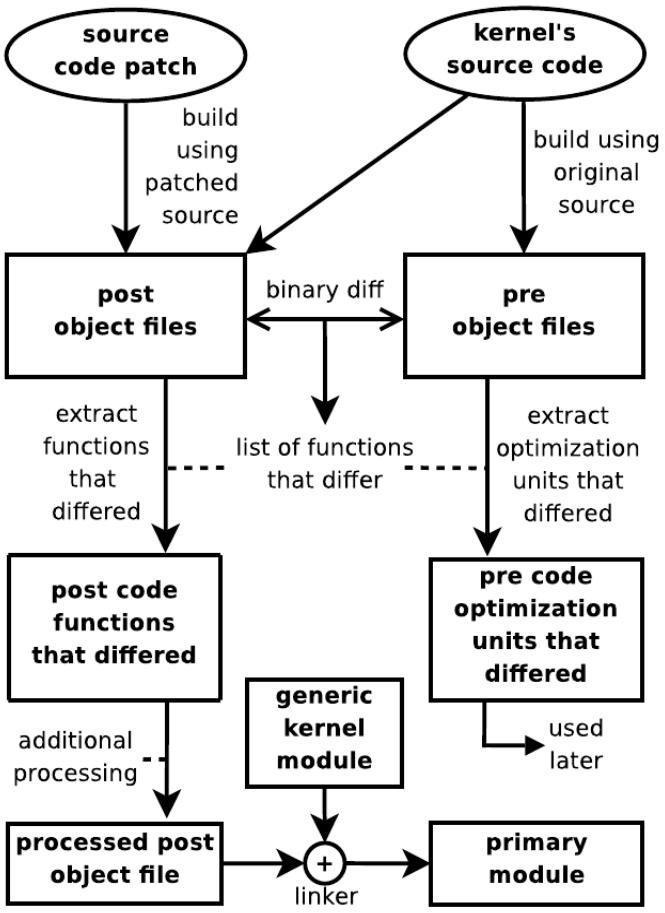

Abbildung 3.1: Ablauf einer Aktualisierung mit *Ksplice* [66]

3.5.3 Besonderheiten beim Compilieren des Kerns

Bei dem Übersetzen des Quellcodes in den Binärcode arbeitet jeder Compiler ein wenig anders und es können abweichende Ergebnisse entstehen. Dies kann an gesetzten Parametern, an Optimierungsalgorithmen oder an der unterschiedlichen Implementierung des Compilers liegen. Wenn *Ksplice* den Binärcode der zwei Kerne vergleicht, müssen nicht exakt die gleiche Compiler-Version oder Compiler-Optionen benutzt werden. Dies ist jedoch zu empfehlen, da so die Wahrscheinlichkeit sinkt, unerwartete Differenzen in den Binärcode-Dateien zu finden. Das Update muss abgebrochen werden, wenn es nicht gelingt, alle Unterschiede in den Binärdateien richtig aufzulösen. [70]

Die veränderten Funktionen sind im Binärcode schwierig aufzufinden und voneinander zu trennen. Um dies zu verbessern, bedient sich *Ksplice* den Compiler-Optionen `-ffunction-sections` bzw. `-fdata-sections`. Mit diesen Parametern entsteht zwar mehr Binärcode, aber der Compiler erzeugt für jede Funktion und Datenstruktur einen eigenen

[70] *Vgl. Arnold, Jeff / Kaashoek Frans: Ksplice: Automatic Rebootless Kern Updates, S. 2-8, Nuremberg 2009.* [66]

Abschnitt. [71] Das erleichtert das Auffinden und Austauschen von Funktionen. [72]

3.6 Stabilität von Ksplice

Die Nutzung von *Ksplice* birgt ein Risiko für die Systemstabilität. Die Entwickler des Systems, *Jeff Arnold* und *Frans Kaashoek*, wiesen darauf bereits im Jahr 2009 in ihrer Untersuchung mit dem Titel „Ksplice: Automatic Rebootless Kernel Updates“ hin. Diese sagt unter anderem aus, dass *Ksplice* nur für kleine Patches und Bugfixes vorgesehen ist. Die meisten Standard-Patches können in ein *Ksplice*-Update umgewandelt werden. Jedoch bedarf es bei einigen einer Modifizierung durch den Programmierer. *Ksplice* kommt nicht mit Änderungen an Datenstrukturen zurecht. Dies muss in einem Patch berücksichtigt und entsprechend angepasst werden. [72]

Oracle umgeht dieses Problem, indem es die Patches für *Ksplice* selbst bereitstellt und diese testet, so dass keine Schäden am Kern verursacht werden. Dadurch wird der Administrator aus der Verantwortung genommen und die Fehleranfälligkeit von *Ksplice* stark reduziert.

[71] *Vgl. Free Software Foundation, Inc.(Hrsg.): 3.10 Options That Control Optimization, 2015.* [68]

[72] *Vgl. Arnold, Jeff / Kaashoek Frans: Ksplice: Automatic Rebootless Kernel Updates, S. 2-8, Nuremberg 2009.* [66]

4 Systemupdates mit Paketmanagern

Das im letzten Kapitel betrachtete *Ksplice* kann nicht für das gesamte Betriebssystem genutzt werden. Verschiedene Dienste und Programme können ebenso das Betriebssystem destabilisieren und benötigen Aktualisierungen. Daher muss noch viel Software über den Paketmanager aktualisiert werden. Der Standardpaketmanager in *Oracle-Linux* heißt *Yum*, dieser wird in diesem Abschnitt näher betrachtet. Zu Beginn wird auf dessen Geschichte eingegangen und seine Funktionsweise erklärt. Es werden einige Verwendungsmöglichkeiten angesprochen und der Aktualisierungsvorgang wird für verschiedene Dateien genauer beschrieben. Am Ende dieses Kapitels wird zusammengefasst, ob und wie ein Paketmanager die Stabilität eines Betriebssystems gefährden kann.

4.1 Yellowdog Updater, Modified

Yum ist die Abkürzung für *Yellowdog Updater, Modified*, er wird unter anderem auf den Distributionen *Red Hat Linux*, *CentOS* sowie auf, dem in dieser Arbeit fokussierten, *Oracle-Linux* eingesetzt. [73] Seinen Ursprung hat der Paketmanager in der Distribution *Yellowdog-Linux*. Dort wurde er unter dem Namen *YUP* (Yellowdog Updater) von *Dan Burcaw*, *Bryan Stillwell*, *Stephen Edie* und *Troy Bengegerdes* programmiert. Später wurde *Yup* weiterentwickelt und zu *Yum* umbenannt. Im Kern ist *Yum* ein Open-Source-Projekt und wurde von vielen Personen weiterentwickelt. Die Initiatoren waren *Seth Vidal* and *Michael Stenner*, ihrerzeit an der „Duke University“ in North Carolina (USA). Später arbeitete *Seth Vidal* bei *Red Hat*, wo er die Weiterentwicklung vorantrieb. [74]

Der Paketmanager *Yum* ermöglicht das Suchen, Installieren, Löschen und Aktualisieren von *RPM*-Paketen. Dabei baut er auf *RPM* auf und ergänzt dessen Funktionsumfang. So ist es mit normalen *RPM*-Befehlen nicht möglich, Abhängigkeiten zwischen einzelnen Paketen automatisch aufzulösen, eine Funktion die allerdings *Yum* beherrscht. Des Weiteren lädt *RPM* die Pakete nicht selbstständig aus einem Repository herunter, diese müssen vor der Installation oder dem Update auf das System geladen werden. [75] Auch an dieser Stelle ist *Yum* einen Schritt weiter und kann sich zu einem oder mehreren Servern verbinden und von dort Pakete beziehen. [76]

[73] *Vgl.Turmbull, James u.a.: Pro Linux System Administration, S. 290 , Berkeley 2009.* [69]
[74] *Vgl.Brown, Robert / Pickard, Jonathan: Yum (Yellowdog Updater, Modified) HOWTO, 2003.* [70]
[75] *Vgl.Turmbull, James u.a.: Pro Linux System Administration, S. 299 , Berkeley 2009.* [69]
[76] *Vgl.Brown, Robert / Pickard, Jonathan: Yum (Yellowdog Updater, Modified) HOWTO, Introduction, 2003.* [70]

Der Paketmanager kann über die Konsole bedient werden. Die meisten Distributionen bieten aber eine grafische Oberfläche an. Zusätzlich existieren auch distributionsunabhängige GUIs (Graphical User Interface), zum Beispiel der *Yum Extender*. [77]

Zwei weitere Funktionen sind das Erstellen eines eigenen *Yum*-Repository und das Automatisieren von Updates. Der letzte Punkt ist eine gute Maßnahme, um die Sicherheit und Stabilität von Computersystemen zu gewährleisten. Bei einer täglichen Prüfung und eventueller Durchführung von Updates sind die Systeme innerhalb von 24 Stunden nach Veröffentlichung des Updates aktualisiert. [78] Jedoch sollten Updates vor jedem Einspielen getestet werden. Selbst wenn der Paket-Entwickler seine Pakete testet, kann es zu Fehlern kommen. Es ist möglich, dass doch ein Fehler in einem Paket vorkommt oder die Kombination von Software nicht mit dem Update kompatibel ist.

4.1.1 Funktionsweise Yum

In diesem Abschnitt wird die Funktionsweise von *Yum* am Beispiel eines Updates beschrieben. Dafür werden die Kommandos der Konsole genutzt, so dass der Ablauf besser unterteilt werden kann.

Im ersten Schritt wird geprüft, ob es Updates für die installierten Pakete gibt. Dabei werden zuerst die Header-Dateien der Pakete heruntergeladen. Diese sind in einer komprimierten XML-Struktur separat auf dem Repository abgelegt. Der *Yum*-Client aktualisiert die Header-Dateien regelmäßig und führt eine Zwischenspeicherung durch. Um diesen Vorgang per Hand zu starten, wird der Befehl `yum check-update` genutzt. [79] Die heruntergeladenen Header-Informationen werden mit denen in der lokalen Paketdatenbank verglichen. Wenn dabei eine neuere Version eines Paketes festgestellt wird, informiert das System den Nutzer über mögliche Updates.

Im nächsten Schritt lädt *Yum* die Pakete herunter, die aktualisiert werden sollen. Alle verfügbaren Updates werden mit dem Kommando `yum update` installiert. Zum Aktualisieren von einzelnen Paketen wird der Paketname angefügt, z.B.: `yum update kile nmap`.[79] *Yum* löst die Abhängigkeiten zwischen den Paketen auf und installiert die entsprechenden Pakete mit. Der Vorgang wird bei solchen Paketen abgebrochen, bei denen es Paketkonflikte, Abhängigkeitsschleifen oder Versionierungskonflikte gibt. [80] Wie die Installation im Detail durchgeführt wird, ist im Abschnitt 4.2 auf Seite 31 näher beschrieben. [81]

[77] *Vgl.Lauridsen, Tim : Yum Extender, About, 2014.* [71]

[78] *Vgl.Brown, Robert / Pickard, Jonathan: Yum (Yellowdog Updater, Modified) HOWTO, What Yum Can Do, 2003.* [70]

[79] *Das Kommando `man yum` öffnet die Anleitung zu Yum samt Befehlsliste.*

[80] *Vgl.Brown, Robert / Pickard, Jonathan: Yum (Yellowdog Updater, Modified) HOWTO, How Yum Works, 2003.* [70]

[81] *Vgl.OMcCallum, Ethan: Managing RPM-Based Systems with Kickstart and Yum, Before You Start, Sebastopol Calif. 2007.* [72]

Die Konfiguration von *Yum* ist in der Datei `/etc/yum.conf` vorzunehmen. Dort werden auch die Repositories konfiguriert. Es ist aber auch möglich einzelne Kopnfigurationsdateien anzulegen Diese befinden sich dann in einem extra Ordner und haben den Suffix *.repo, wie in diesem Beispiel: `/etc/yum.repos.d/public-yum-ol6.repo`. [82] In dem folgenden Auszug ist die Konfiguration des Servers zu sehen, der die neuesten Pakete für *Oracle-Linux 6* vorrätig hält. Hier wird auch der Pfad zum GPG-Schlüssel (GNU Privacy Guard) festgelegt.

```
[public_ol6_latest]
name=Oracle Linux $releasever Latest ($basearch)
baseurl=http://public-yum.oracle.com/repo/OracleLinux/OL6/latest/$basearch/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
gpgcheck=1
enabled=1
```

4.1.2 Verwendung lokaler Repositories

Manchmal ist es sinnvoll ein eigenes *Yum*-Repository aufzusetzen, denn so können die Updates vorher getestet werden. Das folgende Szenario demonstriert eine mögliche Verwendung von lokalen Repositories.

Der *Yum*-Client, der auf dem Clientsystem läuft, ist so konfiguriert, dass er nur Updates vom lokalen Repository A bezieht, das den Aktualitätszustand: X hat. Ein zweites lokales Repository B, welches sich im gleichen Zustand X befindet, übernimmt nun die gewünschten Aktualisierungen aus dem offiziellen Repository der Distribution. Es besitzt nun den Zustand X'. Ein Test-Client bekommt das Repository B zugewiesen, führt die Updates von diesem durch und wird auf Stabilität und Funktionsfähigkeit geprüft. Wenn der Test-Client ausreichend getestet wurde und die Updates für „sicher“ erachtet werden, kann das Repository A die Updates von Repository B übernehmen und an alle Clients verteilen. In diesem Beispiel kommen zwei lokale Repositories zum Einsatz. Dies hat zwei wesentliche Vorteile. Zum einen steht allen Client-Systemen ein Repository zu Verfügung. Zum anderen ist immer ein Backup des Repositories vorhanden, sollte ein Update inkompatibel sein. [83]

4.2 RPM-Pakete

Wie nun bekannt ist, setzt *Yum* lediglich auf *RPM* auf und nutzt Pakete in diesem Format. Um zu klären, ob es mit *RPM*-Paketen möglich ist, Veränderungen am Programmcode

[82] *Vgl. Oracle Corporation (Hrsg.): Oracle Linux Security Guide for Release 6, 2.2 Yum Configuration, 2015.* [73]

[83] *Vgl.OMcCallum, Ethan: Managing RPM-Based Systems with Kickstart and Yum, Why Would I Want My Own Yum Repo?, Sebastopol Calif. 2007.* [72]

während der Laufzeit vorzunehmen, wird im nächsten Abschnitt das *RPM*-Format betrachtet und beschrieben, wie *RPM*-Pakete installiert werden.

Die Abkürzung *RPM* (RPM-Package-Manager) [84] steht sowohl für den Namen eines Paketformates als auch für einen Paketmanager. *RPM*-Pakete haben die Dateiendung *.rpm. Ihre Namen sind nach einem bestimmten Muster aufgebaut. Jeder Paketbezeichner beginnt mit dem Programmnamen (Paketname), gefolgt von der Versionsnummer. Mit einem Bindestrich ist die Releasenummer abgegrenzt. Diese gibt an, wie oft ein Paket in der entsprechenden Version neu gebaut wurde. Das tritt beispielweise auf, wenn kleine Fehlerkorrekturen eingefügt wurden. Am Ende wird die Hardware-Architektur, für die das Paket vorgesehen ist, angegeben. Diese ist durch einen Punkt abgetrennt. In dem unten stehenden Beispiel liegt das Paket „redhat-menus“ in der Version 14.0.0 vor und wurde drei Mal neu veröffentlicht. [85] Das Paket enthält keinen plattformabhängigen Programmcode und endet daher mit „noarch“. [86]

- rpm-4.8.0-38.el6_6.x86_64
- redhat-menus-14.0.0-3.el6.noarch

RPM-Pakete werden mithilfe einer Datenbank verwaltet, in der Informationen zu allen installierten Paketen gespeichert werden. So kann beispielsweise bei der Deinstallation eines Paketes festgestellt werden, ob dies noch von einem anderen Programm oder Paket benötigt wird. In diesem Fall würde es nicht gelöscht werden. [87] Die Datenbank befindet sich im Verzeichnis `/var/lib/rpm` und ist „ [...] in einem proprietären Format abgespeichert, das nur mit dem rpm-Programm zugänglich ist.“ [88]

4.2.1 Aufbau RPM-Pakete

Ein *RPM*-Paket ist ähnlich wie ein Archiv und besteht aus mehreren Dateien. Jeder Paketersteller entscheidet selbst, je nach Verwendungszweck und Art des Paketes, welche Funktionen sein Paket beinhaltet. Im Wesentlichen besteht ein *RPM*-Paket aus vier Teilen. Diese sind der Header, die Programmdaten in Archivform, eine Signatur, sowie verschiedene Skripte. In Abbildung 4.1 sind die Wurzelverzeichnisse zweier *RPM*-Pakete dargestellt. Darin sind der Header jedes Paketes, die Quelldateien im *cpio*-Format (copy in, copy out), zwei Scripte und ein Unterverzeichnis zu erkennen. Die zwei Scripte im Wurzelverzeichnis sind das Installationsscript (*INSTALL) und das Updatescript (*UPGRADE). Weiterhin sind in *RPM*-Paketen Scripte enthalten, die vor und nach der

[84] *Vgl. Bailey, Edward u.a.: Maximum RPM. Taking the Red Hat Package Manager to the Limit, rpm — RPM Package Manager, Version 1.2, 2000.* [74]
[85] *Vgl. Bailey, Edward u.a.: Maximum RPM. Taking the Red Hat Package Manager to the Limit, RPM File Naming Convention, Version 1.2, 2000.* [74]
[86] *Vgl. Merker, Karsten: RPM-Pakete im Eigenbau, Paket-Typen, 2000.* [75]
[87] *Vgl. Welsh, Matt u.a.: Linux Wegweiser zur Installation & Konfiguration, 3. Auflage, 2000.* [76]
[88] *Kalhammer, Florian: Verwendung des Red Hat Package Managers (RPM), Version 1.102.6, 2002.* [77]

Installation sowie vor und nach der Deinstallation ausgeführt werden. Diese befinden sich im Unterverzeichnis `INFO/SCRIPTS/`. [89]

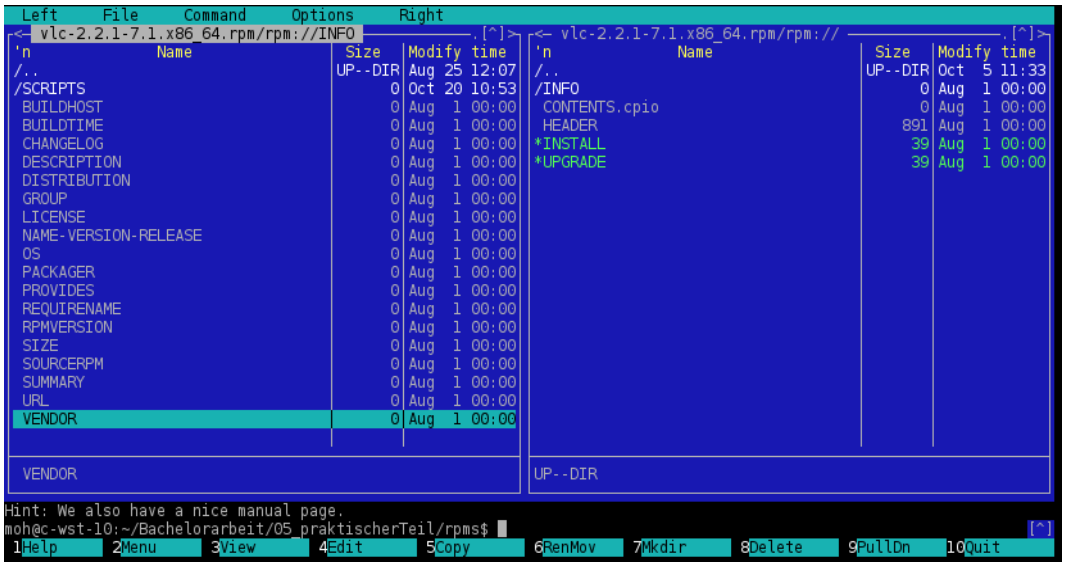

Abbildung 4.1: Struktur von *RPM*-Paketen im *Midnight Commander*

Im Header sind Informationen über das Paket gesammelt. Neben dem Namen, der Paketversion und einer Beschreibung des Paketes, sind Informationen zur Signatur enthalten (siehe Anlage F.1). Weitere Informationen zum Paket finden sich in den Dateien im Verzeichnis `/INFO`. Dabei sind die Dateien `PROVIDES` und `REQUIRENAME` besonders interessant, da diese zur Identifizierung von Abhängigkeiten benötigt werden. In der Datei `PROVIDES` stehen die Bibliotheken und Programmnamen, die das Paket selbst zur Verfügung stellt, wohingegen `REQUIRENAME` die abhängigen Pakete beinhaltet. In einigen Quellen wird statt `REQUIRENAME` der Dateiname `PROVIDES` angegeben, dieser konnte in den Untersuchungen nicht bestätigt werden (siehe Abbildung 4.1). [90] Die eigentlichen Programmdateien liegen in einem Archiv. Unkomprimiert ist es ein gewöhnliches *cpio*-Archiv im *SVR4*-Format (System Five Release 4) mit einer *CRC*-Prüfsumme (Cyclic Redundancy Check) . In Abbildung 4.1 heißen diese `CONTENTS.cpio`. [91] Um die Herkunft von *RPM*-Pakete zu prüfen, sind diese signiert. Damit soll vermieden werden, dass Pakete von Dritten ausgetauscht werden. Denn auf diesem Weg könnte Schadsoftware eingeschleust werden. „Damit diese Überprüfung funktioniert muß eine korrekte Installation von PGP oder GPG vorliegen und die Konfiguration von rpm muß dies berücksichtigen.“ [89]

[89] *Kalhammer, Florian: Verwendung des Red Hat Package Managers (RPM), Version 1.102.6, 2002.* [77]
[90] *Vgl. Merker, Karsten: RPM-Pakete im Eigenbau, Abhängigkeiten und Konflikte, 1999.* [75]
[91] *Vgl. Bailey, Edward u.a.: Maximum RPM. Taking the Red Hat Package Manager to the Limit, RPM File Format, Version 1.2, 2000.* [74]

4.2.2 Paketarten

Dieser Abschnitt unterteilt *RPM*-Pakete in Gruppen. Der Autor möchte auf diesem Weg analysieren, welche Daten bei einem Update übertragen und in das System eingespielt werden.

Grundlegend kann Programmcode in zwei unterschiedliche *RPM*-Pakete verpackt werden. Zum einen sind das die „Binary-RPMs“. Diese sind an die vorgegebene Plattform gebunden und enthalten den bereits kompilierten Programmcode. Wenn das Paket in der entsprechenden Architektur vorliegt, kann es sofort ausgeführt werden. Diese Pakete enden mit dem Suffix `*.rpm`. Die „Source-RPMs“ sind die zweite Variante und beinhalten den Quellcode der Programme. Der Quellcode muss erst noch für die entsprechende Architektur kompiliert werden. Vorteilhaft ist, dass sie so plattformunabhängig sind. Diese Pakete haben den Suffix `*.src.rpm`. [92]

Eine weitere Unterteilung der Pakete kann nach ihrem Inhalt erfolgen. Hier können drei große Gruppen festgelegt werden. Zum einen sind das die Programmpakete. Wie der Name bereits vermuten lässt, beinhalten diese die eigentlichen Programme. Die Größe des Programms ist dabei jedoch unbedeutend, es kann ein simpler Musikplayer oder ein komplexes 3D-Programm sein. Eine weitere Gruppe stellen die Pakete mit Bibliotheken dar. Da sich viele Programme die selbe Bibliothek teilen, ist das Aktualisieren sehr wichtig und zugleich effizient. Wenn ein Sicherheitsupdate einer Bibliothek eingespielt wird, verbessert es die Sicherheit aller Programme, die diese Bibliothek nutzen. Manche Programme nutzen nur bestimmte Versionen einer Bibliothek. So kann es sein, dass bei einem Update mehrere Versionen auf dem System zurückbleiben. In Abschnitt 4.3.2 auf Seite 37 wird auf die unterschiedlichen Bibliotheken näher eingegangen. Als dritte Gruppe sind die Entwicklerpakete zu nennen. Diese beinhalten in ihrem Namen den Zusatz `devel` oder `dev`, was je nach Distribution variiert. Auch wenn Entwicklerpakete teilweise zum Kompilieren von Software benötigt werden, können sie für die Betrachtung in dieser Arbeit weitestgehend vernachlässigt werden. Sie beinhalten Header-Dateien und Quellcode, die für die Weiterentwicklung benötigt werden und sind für Programmierer interessant. [93] Als eine Art von kompilierbaren Paketen wurden Source-RPMs bereits genannt.

4.2.3 Delta-RPM-Pakete

Normalerweise wird bei einem Update das gesamte zu aktualisierende Paket heruntergeladen. Dieses enthält den neuen bzw. veränderten Programmcode, aber auch den aus der bereits installierten Programmversion. Bei kleinen Versionssprüngen, Sicherheitsupdates oder Bugfixes wird meist nur wenig Quellcode verändert, so dass der

[92] *Vgl. Merker, Karsten: RPM-Pakete im Eigenbau, Paket-Typen, 1999.* [75]
[93] *Vgl.Turmbull, James u.a.: Pro Linux System Administration, S. 268, Berkeley 2009.* [69]

überwiegende Paketinhalt bereits auf dem Zielsystem vorhanden ist und somit doppelt übertragen wird. Bei großen Paketen, beispielsweise *LibreOffice*, macht sich das besonders bemerkbar. *Delta-RPM*-Pakete beinhalten dagegen nur die Veränderungen eines Paketes. Mithilfe des Original-Paketes, bzw. aus den bereits installierten Dateien und dem *Delta*-Paket wird ein neues *RPM*-Paket erzeugt. Dieses kann anschließend wie jedes andere *RPM*-Paket installiert werden.

Der größte Vorteil von *Delta*-Paketen ist die Zeiteinsparung bei der Übertragung. Ein Sicherheitsupdate von *Acrobat Reader* ist als *Delta-RPM* nur 2,7 MByte groß, während das komplette *RPM*-Paket ca. 35 MByte groß ist. Dieser Größenvorteil muss sich aber mit entsprechender Rechenleistung und Speicherplatz erkauft werden. Das Erstellen eines *RPM*-Paketes aus einem *Delta-Paket* beansprucht den dreifachen Speicherplatz der unkomprimierten Nutzlast (Payload) im Arbeitsspeicher. [94] *Delta*-Pakete wurden von *Suse* entwickelt, können aber auch in anderen Distributionen wie *Fedora* und *Mandriva* eingesetzt werden. [95] In *Oracle-Linux* unterstützt *Yum* dies standardmäßig erst ab Version 7. In den Versionen 5 und 6 musste zuerst das *Yum-presto-Plugin* installiert werden, um *Delta-RPMs* nutzen zu können. [96]

4.3 Funktionsweise der Updates

Die Installation eines *RPM*-Paketes erfolgt in mehreren Phasen. Dabei werden Scripte in unterschiedlicher Reihenfolge ausgeführt, sofern diese vorhanden sind. Es ist schwer eine Aussage über die Arbeitsweise der einzelnen Pakete zu treffen, da jedes Paket andere Scripte besitzen kann. Das Preinstall-Script trifft Vorbereitungen für die Installation. Das Anlegen von Ordnerstrukturen, das Prüfen von Schreibrechten und das Abfragen von Nutzereingaben könnten darunter fallen. Die Post-Scripte werden für Aufräumarbeiten genutzt, z.B. Verzeichnisse bereinigen oder temporäre Dateien löschen. In Tabelle 4.1 sind die Installationsphasen dargestellt. [97]

Bei einem Update deinstalliert *RPM* zuerst die vorhandene Version des Paketes vom System und installiert anschließend das Paket in der neuen Version. Dabei werden dieselben Mechanismen verwendet wie bei einer Installation und Deinstallation. [98] *RPM* stellt zwei Möglichkeiten zur Durchführung eines Updates zur Verfügung. Die übliche Variante aktualisiert die vorhandene Version des Paketes, installiert das Paket aber auch, wenn es

[94] *Vgl. Hilzinger, Marcel: Effizient Pakete verteilen mit Delta-RPM und Delta-ISO, in Linux Magazin vom 09.2005.* [78]

[95] *Vgl. Kofler, Michael : Linux 2011, Debian, Fedora, openSUSE, Ubuntu, 10. Auflage, S. 515, München u.a. 2011.* [79]

[96] *Vgl. Guillou, Gregory: Fast is the new better, Speeding Up Oracle Linux 7 Updates with „Delta RPMs“, 2014.* [80]

[97] *Vgl. Birnthaler, Thomas / Gottschalk, Hermann: HOWTO zu RPM (RedHat Package Manager), Version 1.14, 2013.* [81]

[98] *Vgl. Kalhammer, Florian: Verwendung des Red Hat Package Managers (RPM), Version 1.102.6, 2002.* [77]

Nr.:	Beschreibung	Typ
1	Preinstall-Script ausführen	Optional
2	Archiv auspacken und Inhalt ins Dateisystem kopieren	Muss
3	Postinstall-Script ausführen	Optional
4	Deinstall-Script ausführen	Optional

Tabelle 4.1: Installationsphasen der *RPM*-Pakete [81]

noch keine vorhandene Version auf dem System gibt (`rpm -U | -upgrade [Optionen] Paketdatei`). Bei der zweiten Variante werden ausschließlich die Dateien aktualisiert, die bereits auf dem System installiert sind (`rpm -F | -freshen [Optionen] Paketdatei`). Für die Untersuchungen in dieser Arbeit spielt die verwendete Option eine weniger große Rolle, da in beiden Varianten immer die auf dem System vorhandenen Daten aktualisiert werden.

In diesem Abschnitt wurde geklärt, dass ein Update nur das Löschen und Kopieren von Daten mit sich bringt und von einer Reihe von Skripten begleitet wird. Bislang kann keine Aussage darüber getroffen werden, wie das System mit Programmdaten umgeht, die gelöscht werden, während das Programm noch im Arbeitsspeicher betrieben wird. Um dies in den nächsten Abschnitten zu untersuchen, werden die unterschiedlichen Programmdateien einzeln betrachtet.

4.3.1 Aktualisieren von Konfigurationsdateien

Ein Teil der zum Programm gehörenden Dateien sind Konfigurationsdateien. In ihnen werden Einstellungen zum Programm vorgenommen. Bei einem Update würden diese Einstellungen verloren gehen und der Nutzer müsste diese neu vornehmen. Der Autor weis aus eigener Erfahrung, dass die Konfigurationsdateien meist einmalig, bei der Einrichtung des Programms oder in großen Zeitabständen bearbeitet werden. Meist ist es umständlich, die genaue Konfiguration wiederherzustellen. *RPM* bietet dafür bereits eine Lösung. Es erstellt drei MD5-Prüfsummen. Durch dessen Vergleich kann festgestellt werden, welche Datei verändert wurde. Eine Prüfsumme wird von der Konfigurationsdatei nach der Installation erstellt (originale Datei). Eine weitere kurz vor ihrem Update (aktuelle Datei) und die letzte Prüfsumme wird von der neuen Konfigurationsdatei (neue Datei) erstellt. Je nach Änderungsszenario entscheidet sich *RPM* für ein entsprechendes Vorgehen. Die Zustandskombinationen, welche die Konfigurationsdateien annehmen können, sind in Tabelle 4.2 aufgelistet. [99] Dieses Vorgehen kommt nicht ohne den Eingriff eines Nutzers aus, denn *RPM* überträgt nicht die Änderungen aus einer alten Konfigurationsdatei in eine neue. [100] Ein Update kann zwar während der Laufzeit des Programms eingespielt werden, welche Konfiguration ab dann genutzt wird, muss der

[99] *Vgl. Bailey, Edward: Maximum RPM. Taking the Red Hat Package Manager to the Limit, Using RPM to Upgrade Packages, Version 1.2, 2000.* [74]

[100] *Vgl. Novell, Inc. (Hrsg.): openSUSE-Dokumentation, 2007.* [82]

Administrator jedoch händisch nacharbeiten.

ori.	akt.	neu	Vorgehen
X	X	X	Alle drei Dateien sind gleich. *RPM* überschreibt die alte Konfigurationsdatei mit der aus dem Update (Veränderungen an den Rechten der Datei können nicht mit MD5 erkannt werden).
X	X	Y	Die Datei im Update unterscheidet sich von den beiden anderen. *RPM* überschreibt die originale Datei. Es wird kein Backup der aktuellen Konfigurationsdatei erstellt, da die aktuelle Datei der originalen gleicht.
X	Y	X	Es wurden Einstellungen vorgenommen. Das Update enthält eine aktuellere Konfigurationsdatei. Die aktuelle Datei wird beibehalten, es wird nichts ersetzt.
X	Y	Y	Die originale Datei wurde so editiert, dass diese der neuen Datei gleicht. Dies kann beispielsweise vorkommen, wenn ein Bugfix händisch vorgenommen wurde, welcher nun auch mit dem Update eingespielt werden soll. Die originale Datei wird durch die neue ersetzt.
X	Y	Z	Alle drei Dateien sind unterschiedlich. Dieses Szenario entsteht, wenn in einem Update eine komplett neue Datei enthält und die originale Datei editiert wurde. Die neue Datei wird aus dem Update übernommen. Zuvor wird die aktuelle Datei unter einem anderen Namenssuffix gesichert. Dieser kann sich in den Distributionen unterscheiden und endet mit dem Suffix `*.rpmsave` oder `*.rpmorig`. [101] Der Nutzer wird über diesen Vorgang benachrichtigt, z.B.: `warning: /etc/skel/.bashrc saved as /etc/skel/.bashrc.rpmsave`. Eine Ausnahme von dieser Regel existiert, wenn im *spec-file* des *RPM*-Paketes die Einstellung `%config(noreplace)` gesetzt ist. Dann wird die aktuelle Datei beibehalten und die neue Konfiguration aus dem Update wird mit dem Suffix `*.rpmnew` dazu gespeichert. [102]

Tabelle 4.2: Zustandskombinationen der Konfigurationsdateien [84]

4.3.2 Aktualisieren von Biblioheken

Bibliotheken sind eine weitere Gruppe von Dateien, die aktualisiert werden. Ihre Austauschbarkeit steht dabei im Mittelpunkt dieser Betrachtung. Daher werden sie im folgenden nach ihren Bindungsarten in statische Bibliotheken, gemeinsame Bibliotheken und

[101] *Vgl. Novell Inc. (Hrsg.): openSUSE-Dokumentation, 2007.* [82]
[102] *Vgl. Warbrick, Jon: RPM, %config, and (noreplace).* [83]

dynamisch ladbare Bibliotheken unterteilt. Grundlegendes zu Bibliotheken ist in Abschnitt 2.3 auf Seite 10 zu finden.

- Statische Bibliotheken werden beim Kompilieren in den Programmcode der Anwendung kopiert. Ein Programm muss erst mit der neuen Bibliotheksversion übersetzt werden, wenn sie aktualisiert wurde, um die Änderungen nutzen zu können. Der neue Code wird also mit den Programmdaten, beim Aktualisieren der Anwendung eingespielt.

- Gemeinsame Bibliotheken werden beim Start eines Programmes an den Programmcode „gelinkt“. Das bedeutet, wenn eine solche Bibliothek bei einem Programmstart, sich noch nicht im Speicher befindet, wird sie geladen und für die Anwendung nutzbar. Dabei bleibt sie so lange im Arbeitsspeicher, bis sie von keinem Programm mehr benötigt wird. Um eine aktualisierte Bibliothek zu laden, muss das Programm neu gestartet werden. Durch dynamische Links können Bibliotheken mit mehreren Namen benannt werden, so „[..] wird beispielsweise die Datei libcrack.so.2.7 auch als libcrack.so.2 und als libcrack.so bekannt gemacht. Ein Programm kann nun gegen libcrack gelinkt werden, und es wird auch noch funktionieren, wenn eine libcrack.so.2.8 installiert wird, oder ein System nur eine libcrack.so.2.6 anbietet. Linkt ein Programm direkt gegen eine Version, so muss natürlich die genau passende vorhanden sein.“ [103] Bei einem Update sind die Programme im „Vorteil“, die nicht gegen den *real name* gelinkt wurden. Denn der *sonem*, bzw. der *linker name* ändern sich nicht so schnell. Dabei ist es auch möglich, mehr als eine Bibliotheksversionen auf dem System zu installieren. Das erhöht zwar die Kompatibilität zu den Programmen, aber nicht die Sicherheit und Stabilität, da die Verbesserungen der neueren Version fehlen. [104] Es sollte trotzdem darauf geachtet werden, dass das Programm die neue Bibliothek unterstützt. Andernfalls können Probleme beim Neustart des Programms auftreten.

- Dynamisch ladbare Bibliotheken ermöglichen es theoretisch Updates während der Laufzeit eines Programmes durchzuführen. Die aktualisierte Bibliothek kann während der Programmausführung geladen werden. Doch genau der Ladevorgang muss genauer betrachtet werden. Das Programm steuert selbst, wann es Bibliotheken lädt und entlädt. Je nach Programmierung und Nutzung der Anwendung kann das zu einem beliebigen Zeitpunkt sein. Sollte das Programm nicht kompatibel mit der neuen Bibliotheksversion sein, kann es zu einem Absturz des Programmes führen. [105] Aus Sicherheitsgründen sollte ein Programmierer die Bibliothek vor dem Laden auf Kompatibilität prüfen und eventuelle Fehler abfangen.

[103] *Steffen Dettmer: Self Linux. Bibliotheken, Version 0.12.3.* [85]
[104] *Vgl. Dettmer, Steffen / Hemm, Torsten: SelfLinux, Bibliotheken, Version 0.12.1.* [25]
[105] *Vgl. Glatz, Eduard: Betriebssysteme. Grundlagen, Konzepte, Systemprogrammierung, S. 594, Heidelberg 2006.* [28]

Wenn Bibliotheken von andere abhängig sind, dann werden sie rekursiv mit geladen. Wobei die Systemfunktion `dlopen()` jede Bibliothek nur einmal in den Arbeitsspeicher lädt. Soll ein zweites mal die selbe geladen werden, dann gibt die Funktion den gleichen Datei-Handler zurück. Dabei wird ein Referenzzähler inkrementiert. Dieser wird nur mit dem Aufruf der Funktion `dlclose()` abgebaut. Erst wenn kein Programm mehr eine Referenz auf die Bibliothek hält wird diese aus den Arbeitsspeicher entladen. Dies gilt für gemeinsame und dynamisch ladbare Bibliotheken. [106]

4.3.3 Aktualisierung von Dateien

Die letzte Gruppe von Daten die betrachtet werden muss ist die der Programmdateien. Während eines Updates werden diese, durch ein Script aus dem Archiv des *RPM*-Paketes, auf die Festplatte kopiert. Die vorher dort vorhandenen Daten werden gelöscht. Bei Linux-Systemen kann der Nutzer „[...] jede Datei jederzeit 'löschen' und durch eine neue Version austauschen. Dabei wird die alte Datei aber nicht wirklich gelöscht, sondern nur im Verzeichnis 'geunlinkt' und verbleibt im System bis der letzte Prozess sie geschlossen hat.“ [107] Wenn bei Schreib- oder Löschoperationen Probleme Auftreten werden diese vom Dateisystem behandelt. Das bei Linux verbreitete Dateisystem *Ext4* bietet, wie auch andere Dateisysteme, verschiedene Funktionen um die Konsistenz von Daten auch im Problemfall zu gewährleisten. Eine Betrachtung der Verbreiteten Dateisysteme und ihre Funktionen ist an dieser Stelle zu Umfangreich und würde sich vom Thema der Arbeit recht weit entfernen. Daher wird dies nicht weiter beschrieben.

4.4 Stabilität und Sicherheitsprobleme mit Yum/RPM

Die theoretische Betrachtung beleuchtete bislang die einzelnen Komponenten und Funktionsweisen in einem Paketmanager und wie mit unterschiedlichen Dateien bei einer Aktualisierung umgegangen wird. Im Hinblick auf die Thematik dieser Arbeit wird nun auf die bekannten Schwächen eingegangen, die eventuell die Stabilität des Betriebssystems gefährden könnten.

Die in einem *RPM*-Paket enthaltenen Scripte können auch destruktiv verwendet werden, im einfachsten Fall mit einer Löschoperation bis hin zur Einschleusung von Schadsoftware. Durch die Signierung der Pakete und die Nutzung von vertrauenswürdigen Quellen wird das Risiko reduziert, aber nicht ausgeschlossen.

In seltenen Fällen entstehen sogenannte Abhängigkeitskonflikte. Diese treten auf, wenn eine Datei nicht nur von einem, sondern von mehreren Paketen zur Verfügung gestellt wird. In der Regel verhindert dies die Distribution, welche die Pakete, bzw. den

[106] *Vgl. Richter, Adam u.a.: dlopen(3) - Linux man page, Release 4.02, 2015 .* [86]
[107] *Hochstätter, Christoph: Enterprise-Linux: Red-Hat-Alternativen im Vergleich, 2012.* [87]

Paket-Server (Repository) pflegt. Selbst gebaute Pakete oder solche aus Fremdquellen (nicht von der Distribution angebotene Pakete) können diesen Fehler provozieren. Dadurch kann die Integrität des Systems und die Updatefähigkeit gefährdet werden.

Bei der Installation von Bibliotheken besteht „[...] das Problem, dass eine neue symbolische Verknüpfung auf eine nicht korrekte rückwärtskompatible Version dazu führen kann, dass vorhandene Applikationen nicht mehr sauber laufen.“ [108]

4.5 Fazit

Der Paketmanager *Yum* bietet selbst keine Möglichkeit zum Live-Patching. Auch *RPM* verfügt nicht über eine solche Funktion. Das Austauschen von Dateien während der Programmlaufzeit ist unter Linux standardmäßig möglich und somit auch ein Updatevorgang. Trotzdem muss das Laden der geänderten Funktionen in den Arbeitsspeicher noch mit einem Neustart der Anwendung, bzw. dem Nachladen der Bibliothek erfolgen.

Dabei ist noch unklar, welche Version einer Bibliothek geladen wird, wenn sie während der Laufzeit aktualisiert wurde. Es konnte nicht recherchiert werden wie sich die Funktion dlopen() verhält. wenn sie als Parameter zum Laden einer Bibliothek, deren `soname` oder `linker name` übergeben bekommt. Erkennt die Funktion die aktualisierte Bibliothek und lädt diese in den Arbeitsspeicher oder gibt die Funktion eine weitere Referenz auf die im Speicher vorhandene, inaktuellen Version? Hat das Auswirkungen auf die Stabilität von Programmen oder des Betriebssystems? Um diese Fragen zu klären, werden im folgendem Kapitel entsprechende Tests durchgeführt.

An verschiedenen Stellen des Aktualisierungsprozesses kann das System destabilisiert werden. Diese Probleme sind jedoch schon länger bekannt und es existieren Vorkehrungen und Verhaltensregeln, die diese Risiken stark verringern. Diese beziehen sich aber auf allgemeine Updates und nicht auf das Aktualisieren während der Laufzeit.

[108] *Glatz, Eduard: Betriebssysteme. Grundlagen, Konzepte, Systemprogrammierung, S. 594, Heidelberg 2006.* [28]

5 Praktische Untersuchungen zu Ksplice und Yum

Bisher wurde die Funktionsweise von *Ksplice* und *Yum* theoretisch erörtert und mögliche Probleme in Bezug auf die Systemstabilität aufgezeigt. Dabei konnten nicht alle Fragen geklärt werden. Beispielsweise ist nicht sicher, wie sich das Austauschen von Bibliotheken während der Laufzeit auf Programme auswirkt und ob eine Aktualisierung des Kerns in einem belasteten System erfolgreich ist. Eine dynamische Analyse, mit praktischen Tests, soll in diesem Kapitel zu einer Beantwortung der offenen Fragen führen.

Details zu den einzelnen Testdurchläufen sind dem Anhang dieser Arbeit zu entnehmen, die Auswertungen werden in diesem Kapitel vorgenommen. Zu Beginn wird das Testsystem vorgestellt und mit einem Leerlauftest auf seine Funktionsfähigkeit überprüft. Dieser erste Test stellt den Ist-Zustand fest. Dafür werden die Log-Daten des Betriebssystems und der Datenbank ausgewertet und die Stabilität des Systems fest gestellt. Durch einen Snapshot wird das System vor jedem neuen Test wieder auf seinen diese Ist-Zustand zurückgesetzt.

Die zu erwartenden Ergebnisse jedes Test stellen eine Hypothese dar, die am Ende der Untersuchung durch eine Ist/Soll-Analyse verifiziert oder falsifiziert wird. Dies geschieht,am Ende der Testauswertung, in tabellarischer Form. Im Sinne dieser Arbeit definiert sich der Ist-Zustand als ein Computersystem, dessen Softwarekomponenten eine genaue Version haben. Das selbe Computersystem, mit einer aktuelleren Version ausgewählter Software, stellt den Soll-Zustand dar. In den praktischen Tests wird durch eine Aktualisierung von Programmcode und Bibliotheken, in den Soll-Zustand übergegangen. Dabei muss das Computersystem in beiden Zuständen und während des Zustandswechsel stabil funktionieren. Dies ist die stabile Eigenschaft des Testsystems (siehe Kapitel 2.2). Sie wird durch die Begutachtung von Log-Dateien festgestellt und dient zur Auswertung der Hypothese.

Nach dem Leerlauftest wird untersucht, ob eine Software mit mehreren Versionen einer Bibliothek stabil funktioniert. Als zu untersuchende Software wird eine *Oracle*-Datenbank herangezogen. Alle durchgeführten Tests sind Makrotests, d.h. es wird nicht eine einzelne Komponenten, wie der Hauptprozessor, sondern das gesamte System getestet. Dies bedingt das Thema, da die Stabilität des Betriebssystems mit analysiert werden muss. Die Tests enden mit den Untersuchungen zur *Ksplice*. Das Kapitel schließt mit einer Auswertung aller Test und der darin gewonnen Erkenntnisse ab.

5.1 Testsysteme und Konfiguration

Für die Durchführung der praktischen Tests wurden zwei virtuelle Maschinen mit *Oracle-Linux 6* und *7*, jeweils mit einer *Oracle*-Datenbanken genutzt. Die Testsysteme wurden nicht vom Autor eingerichtet, sondern vom betreuenden Unternehmen zur Verfügung gestellt. Die Datenbanken setzen auf einer ASM-Grid-Infrastruktur auf. Diese Struktur wird in der Praxis verwendet um die Ressourcenverwaltung der Datenbank skalierbar und damit flexibel zu halten. Die Testsysteme sollen möglichst praxisnahe Untersuchungen erlauben. Die Datenbank selber stellt als Serveranwendung ein gängiges Anwendungsbeispiel für *Oracle-Linux* dar. Die einzeln startenden und stoppenden Datenbankprozesse eignen sich gut für die Untersuchung von dynamischen Bibliotheken. Die Details zu den Testsystemen sind in Tabelle 5.1 aufgelistet. In einzelnen Untersuchungen wurde der Linux-Kern der Testsysteme ausgetauscht, dies ist den Protokollen der entsprechenden Tests zu entnehmen.

Testsystem A:	
System	Virtuelle Maschine VMX-10 *VMware* 5.5
Prozessor	2x vCPU
Arbeitsspeicher	8 GByte
Distribution	*Oracle-Linux 6*
zu testende Software	*Oracle-Linux*, Orapdflatexcle-Datenbank 11.2.0g, mit ASM-Grid-Infrastruktur 12.1.0.2.0

Testsystem B:	
System	Virtuelle Maschine VMX-10 *VMware* 5.5
Prozessor	2x vCPU
Arbeitsspeicher	4 GByte
Distribution	*Oracle-Linux 7*
zu testende Software	*Oracle-Linux*, *Oracle*-Datenbank 12c, mit ASM-Grid-Infrastruktur 12.1.0.2.0

Tabelle 5.1: Übersicht der Testsysteme

5.1.1 Verwendete Testsoftware

Zur Analyse werden die unter Linux üblichen Werkzeuge eingesetzt, diese sind in den meisten Distributionen vorhanden. Dazu zählen unter anderem *Midnight Commander*, *lsof*, *strings* und *ps*. Zusätzlich wurde der Lastgenerator *Swingbench* (Version 2.5) und das Analysewerkzeug *Strace* (Version 4.8-10) genutzt. Die Aufgaben der Softwares sind in Tabelle 5.2 aufgelistet.

Um Last auf den Testsystemen zu generieren wurde *Swingbench* eingesetzt. Diese von Dominic Giles entwickelte Software, simuliert die Verwendung einer Datenbank. Im

Gegensatz zu *Oracle Real Application Testing*, eine von *Oracle* angebotene Analyse-Software für Datenbanken, ist *Swingbench* kostenlos.

Als Datenbankanalyse-Software wird mit ihr in erster Linie die Datenbank getestet. Die Testauslastung kann dabei so hoch gesetzt werden, dass sie indirekt das gesamte System belastet. Daher wird die Software auch für die Untersuchungen mit *Ksplice* genutzt. Viele Lastgeneratoren sind für diese Arbeit nicht geeignet, denn diese fokussieren sich auf einzelne Komponenten, wie Prozessorlast, Schreiboperationen oder den Arbeitsspeicher. Eine Testsoftware welche speziell den Linux-Kern testet konnte jedoch nicht recherchiert werden. Zusätzlich zur Simulation wertet *Swingbench* die Prozessorauslastung, Transaktionsrate und die Schreiboperationen des Testsystems aus. Als Java-Anwendung ist sie ohne Installation, sofort nach dem Herunterladen, einsetzbar. *Swingbench* verbindet sich von einem anderen Computer auf das Testsystem, so verfälscht es durch seine eigenes Verhalten nicht die Tests. [109]

Programm	Funktion
lsof	Zeigt an, welche Dateien ein Prozess geöffnet hat.
Strace	Nachverfolgung von Systemaufrufen und Signalen.
Swingbench	Simuliert die Verwendung der Datenbank.
Yum	Paketmanager, der Aktualisierungen in das System einspielt.
ps	Listet auf dem System ausgeführte Prozesse auf.
Midnight Commander	Dateibrowser für die Konsole.

Tabelle 5.2: Kurzbeschreibung der Werkzeuge

5.1.2 Konfiguration von Swingbench

Für die Lastsimulation bringt *Swingbench* drei Test-Vorlagen mit, die sich in der Verteilung ihrer Transaktionen unterscheiden. Diese Vorlagen beinhalten einen Datenbestand, der zuvor in die Datenbank eingespielt werden muss. Dazu gibt es im Verzeichnis von *Swingbench* unter `swingbench/bin/` folgende drei Setupdateien:

1. oewizard (Order Entry)
2. ccwizard (CallingCircle)
3. shwizard (Sales History)

Für die Tests wurde die „Order Entry“-Vorlage genutzt, da bei ihr die Transaktionen am gleichmäßigsten verteilt sind. Bei der „CallingCircle“-Vorlage überwiegt der Teil mit `Select`-Abfragen (83%). Dies ist für diese Untersuchung nicht zweckmäßig, da ein möglichst breites Feld von Funktionen verwendet werden soll. Die Stabilität ist nicht nur bei der üblichen Nutzung gefährdet, sondern auch durch Sonderfälle. Der Zeitumfang

[109] *Vgl. Giles, Dominic: dominicgiles.com, 2010.* [88]

reicht nicht aus, um alle möglichen Grenzfälle von *Oracle*-Datenbanken abzudecken. Es wird daher versucht, die verwendeten Funktionen durch Ergänzung von weiteren Transaktionen und das Wählen einer geeigneten Vorlage zu optimieren. Bei der „Order Entry“-Vorlage werden keine `Delete` Statements ausgeführt. Diese wurden zusätzlich hinzugefügt. Die verwendeten Transaktionen können der Tabelle 5.3 entnommen werden. Die „Load-Ratio“ gibt das Verteilungsmuster der Transaktionen an, sie ist aber nicht als Prozentsatz anzusehen. Die von der Testvorlage übernommen Transaktionen haben fest vorgegeben Load-Ratios. [110]

Transaktionsbezeichnung	**Load-Ratio**
../.stresstest.StressTestDelete	50
../.plsqltransactions.BrowseAndUpdateOrders	5
../.plsqltransactions.ProcessOrders	5
../.plsqltransactions.NewOrderProcess	40
../.plsqltransactions.BrowseProducts	50
../.plsqltransactions.UpdateCustomerDetailsV2	10
../.plsqltransactions.NewCustomerProcessV2	15

Tabelle 5.3: Übersicht der *Swingbench*-Transaktionen

Um eine gleichmäßige Transaktionsrate zur Datenbank aufzubauen, verbindet sich *Swingbench* mit zehn Nutzern und löst die Verbindungen erst am Ende des jeweiligen Tests. Eine höhere Anzahl an Nutzern veranlasst die Datenbank, in kürzeren Intervallen die Änderung aus dem Arbeitsspeicher auf die Festplatte zu schreiben. Dies hat zur Folge, dass die Rate der verarbeiteten Transaktionen niedriger ist als bei einer kleineren Nutzeranzahl. Eine höhere Transaktionsrate belastet das Testsystem mehr und liefert ein geeigneteres Testergebnis.

Auf das regelmäßige Ein- und Ausloggen der Nutzer während des Testlaufs wurde verzichtet, da dies Probleme bei der Verwendung von *Swingbench* auslösen konnte. Außerdem war es nur schwer nachvollziehbar welche Prozesse dadurch gestartet und gestoppt würden. Wenn zu viele Prozesse geändert hätten, würden diese bei der Registrierung in *Strace* fehlen. Zudem schwankt der Leistungs- und Transaktionsgraph sehr, wodurch andere Unregelmäßigkeiten nicht erkannt würden.

In Anlage F.5 sind vier Screenshots zu sehen, auf denen die Leistungsgraphen von *Swingbench* abgebildet sind. Von oben nach unten sinkt die getestete Nutzerzahl von Hundert über Fünfzig und Fünfundzwanzig, bis auf Zehn. An den Transaktionen und den Lese-Schreib-Operationen ist zu erkennen, dass eine gleichmäßige Transaktionsrate zwischen Zehn und Fünfundzwanzig Nutzern erreicht werden konnte. Bei einer höheren Nutzeranzahl kommt es zudem zu Fehlermeldungen in *Swingbench* und es werden nicht alle vorgegebenen Nutzer an der Datenbank angemeldet. Folglich werden die Tests mit Nutzerzahlen zwischen Zehn und Fünfundzwanzig durchgeführt.

[110] *Vgl. Giles, Dominic: SwingBench, Reference and User Guide, S. 21 f 2002.* [89]

5.1.3 Konfiguration der Aufzeichnungsprogramme

Das Programm *Strace* wurde standardmäßig mit den Parametern `-f -tt -e open -o /Pfad/zur/Logdatei.log -p PID` ausgeführt. Der Parameter `-p PID` musste dabei für jeden zu überwachenden Prozess einzeln angegeben werden. Die verwendeten Parameter haben folgende Bedeutung.

- `-f` - Zeichnet auch Informationen zu den gestarteten Kindprozessen auf.
- `-tt` - Fügt zu jeden Eintrag eine Zeitangabe hinzu.
- `-e EVENT` - Filterung der aufzuzeichnenden Systemaufrufe (Systemcalls).
- `-o /Pfad/zur/Logdatei.log` - Schreibt die Informationen in eine Datei.
- `-p PID` - Gibt den Prozess an der überwacht werden soll.

Der Befehl *lsof* wird mit dem Parameter `-c oracle` ausgeführt, um geöffnete Dateien der *Oracle*-Datenbank anzuzeigen. Zusätzlich wird, mit Hilfe von `| grep .so`, nur nach geöffneten Bibliotheken gefiltert, bevor die Ausgabe des Befehls in eine separate Log-Datei gespeichert wird. Abweichungen und Variationen bei der Verwendung von Parametern befinden sich in den entsprechenden Anlagen der Test.

5.1.4 Konfiguration des Logsystems

Die Systemaufzeichnungen (Syslogs) wurden für die Tests angepasst, um diese einfacher auswerten zu können. Dazu wurden folgende Einträge in die Datei `/etc/rsyslog.conf` hinzugefügt.

- `user.warn /home/oracle/logs/program.log`
  Schreibt alle Nachrichten aus normalen Anwenderprogrammen in die Datei „program.log“, ab der Priorisierung „Warnung“.
- `daemon.warn;syslog.warn /home/oracle/logs/dienste.log`
  Schreibt alle Nachrichten aus allen Diensten in die Datei „dienste.log“, ab der Priorisierung „Warnung“.
- `kern.warn /home/oracle/logs/kern.log`
  Schreibt alle Nachrichten, die den Kern betreffen, in die Datei „kern.log“, ab der Priorisierung „Warnung“.
- `kern.warn;authpriv.warn;auth.warn;daemon.warn;syslog.warn;user.warn; /home/oracle/all.log`
  Zusammenfassung aller Nachrichten, ab der Priorisierung „Warnung“.

Die Priorisierung „Warnung“ enthält Warnungen und alle Nachrichtenprioritäten darüber (warning, error, critical, alert, emergency).

Die Datenbank und die ASM-Grid-Struktur erstellen ihre eigenen Log-Dateien in entsprechenden Verzeichnissen. Diese werden zur Einschätzung der Stabilität auf ungewöhnliche Einträge hin analysiert.

- Log des ASM-Grid:
  /u01/app/oracle/diag/asm/+asm/+ASM/trace/alert_+ASM.log
- Logs der Datenbank:
  /u01/app/oracle/diag/rdbms/orcl/orcl/trace/alert_orcl.log
  /u01/app/oracle/log/diag/asmcmd/user_oracle/c-moh-ksplice-01.
  test.aspicon.lan/alert/alert.log

Die in den Tests aufgezeichneten Informationen wurden abgespeichert und befinden sich auf dem, der Arbeit beigefügtem, Datenträger.

5.2 Leerlauftest / Basistest

Um eine Vergleichsbasis zu erstellen, wurden die beiden Testsysteme Fünfzehn Stunden lang unter Last betrieben. Dabei wurde unter Zuhilfenahme von *Swingbench* die Verwendung der Datenbank simuliert. So sollte sicher gestellt werden, dass die Systeme nicht bereits vor den Tests einen Fehler aufweisen. Die ermittelten Werte können für Vergleiche herangezogen werden. Mit dem Basistest (Leerlauftest) wurden die hier aufgelisteten Ziele verfolgt:

1. Untersuchen, ob das Betriebssystem und die darauf laufende Datenbank stabil funktioniert.
2. Schaffung einer Vergleichsbasis durch Beobachtung und Aufzeichnung des Systemverhaltens.

5.2.1 Durchführung

Bei der Durchführung dieses Tests wurden keine Aktualisierungen vorgenommen. Die verwendeten Softwares wurden wie folgt konfiguriert.

- Die Software *Swingbench* meldete zehn Nutzer an der Datenbank an und erzeugt für Fünfzehn Stunden Last. Des Weiteren wurde die, für diese Arbeit, übliche Standardkonfiguration gewählt. Siehe dazu Kapitel 5.1.2.
- Die Software *Strace* wurde zu Beginn mit den Parametern
  `-f -tt -o /Pfad/zur/Logdatei.log -p PID` betrieben, um möglichst viele Vergleichsinformationen aufzuzeichnen. Über die Testdauer von Fünfzehn Stunden wurde jedoch der Datenträger des Testsystems durch die Log-Dateien von *Strace* vollgeschrieben. In den nachfolgenden Tests wurde daher der Parameter `-e open`

ergänzt. Er veranlasst *Strace* dazu Systemaufrufe des Typs `open` aufzuzeichnen. Das reduziert die Größe der Log-Dateien.

- Die Software *lsof* wurde wie in Kapitel 5.1.3 verwendet.

Die Datenbank befindet sich zu Beginn des Tests im hochgefahrenem Zustand, um so möglichst viele Prozesse in *Strace* eintragen zu können. Nach dem Beginn der *Strace*-Aufzeichnungen wurde *Swingbench* gestartet, um Last auf der Datenbank zu generieren. Im Anschluss erzeugte *lsof* ein Abbild der geöffneten Bibliotheken. Das Testsystem wurde danach Fünfzehn Stunden lang arbeiten gelassen ohne ein Update durchzuführen. Sollte das problemlos funktioniert haben, konnten die erstellten Log-Dateien des Systems und der Datenbank ausgewertet werden. Die Abbildung 5.1 verdeutlicht das Vorgehen nochmal.

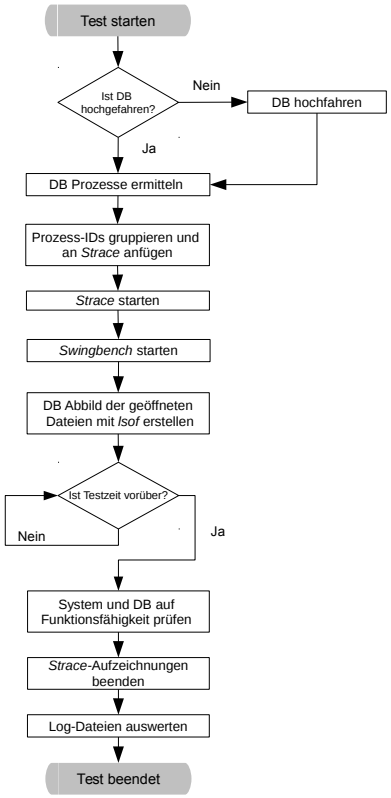

Abbildung 5.1: Durchführung der Basis-Tests

5.2.2 Auswertung

Die Tests liefen ohne feststellbare Fehler durch. Die Testsysteme können demnach für die Untersuchungen verwendet werden. Details zu den Test der beiden Systeme sind in Anlage A.1 auf Seite 77 zu finden. Die Log-Dateien des Systems enthielten keine Einträge. Auch durch das zusätzliche Überprüfen der Dateien in `/var/log/` konnten keine Probleme im System festgestellt werden. Ebenso fanden sich in den Aufzeichnungen von

Strace keine erkennbaren Fehler. Die Systeme reagierte nach dem Ende der Testzeit angemessen und die Datenbank verarbeitete Aufgaben.

Die Aufzeichnungen von *Strace* wurden bei dem Test für das Testsystem A mehrere Gigabyte groß, daher wurde der Inhalt dieser Dateien nach den Zeichenketten „open“ und „.so“ gefiltert und in die Ausgabe in „straceOracleHomeProVergGefiltert.log“ und „straceHintergrundProVergGefiltert.log“ gespeichert. Nur die gefilterten Dateien wurden aufbewahrt. Die Ergebnisse er Test sind nochmal in Tabelle 5.4 aufgelistet.

Erwartetes Ergebnis	**Tatsächliches Ergebnis**
Datenbank funktioniert stabil (15 Stunden Testzeit).	wurde erfüllt
System funktioniert stabil (15 Stunden Testzeit).	wurde erfüllt
keine auffälligen Einträge in den Systemlogs	wurde erfüllt
keine auffälligen Einträge in den Datenbanklogs	wurde erfüllt

Tabelle 5.4: Ergebnisse Basis-Tests

5.3 Voruntersuchung - Bibliotheken eines Datenbankprozesses

Diese Untersuchung wird in Vorbereitung zu dem Test 5.5 auf Seite 52 durchgeführt. Dafür müssen System-Bibliotheken bestimmt werden, welche die Datenbank nutzt. Weiterhin ist der Ladezeitpunkt der Bibliotheken von Interesse, da so bestimmt werden kann, um welche Bibliotheksart es sich handelt. Dabei wäre es zielführend, eine Bibliothek zu ermitteln deren Ladevorgang zu einem bestimmbaren Zeitpunkt wiederholbar ist. Mit ihr könnte überprüft werden, ob die Datenbank mit mehreren Bibliotheks-Versionen stabil betrieben werden kann. Zur Datenbankinstallation gehört ein Paket, welches die Abhängigkeiten der Datenbank installiert. Mit einer Untersuchung dieses Installationspaketes könnten einige Bibliotheken ermittelt werden, welche die Datenbank verwendet. Dadurch kann aber nicht festgestellt werden, ob diese von der Datenbank wirklich benutzt werden oder nur zu dem Paket gehören. Daher ist es sinnvoller die Datenbankprozesse zu untersuchen und zu überprüfen, welche Bibliotheken von ihnen geladen werden. Hierfür kann unterschiedlich vorgegangen werden. Entweder werden alle Prozesse auf einmal überprüft und die ermittelten Bibliotheken werden im Anschluss den Prozessen zugeordnet oder einzelne Prozesse werden untersucht. Für das zweit genannte Vorgehen ist der Nutzerprozess der Datenbank geeignet, da er eine zentrale Rolle bei der Transaktionsverarbeitung hat und durch das Anmelden eines Nutzers an der Datenbank gestartet wird. So kann dessen Start- und Endzeitpunkt bei einem Test gesteuert werden. Folgend wird beschrieben, wie ein Nutzerprozess auf das Laden von dynamischen Bibliotheken hin untersucht wurde.

5.3.1 Durchführung

Der Test begann mit *Swingbench*. Das Programm meldete einen Nutzer an der Datenbank an, wodurch ein Nutzerprozess startete. Das Kommando `ps` wurde so auf dem Testsystem ausgeführt, dass es eine Liste von laufenden Nutzerprozessen der Datenbank anzeigt. Dabei informiert eine Spalte in der Ergebnisausgabe darüber, wie lange die Prozesse schon ausgeführt werden:

```
[root@c-moh-ksplice-01  ]# ps -eo cmd,''%p %t''-sort pid | grep oracleorcl
oracleorcl (LOCAL=NO) cmd, 7161  00:05
```

Anhand der Zeit-Spalte lässt sich der zuletzt gestartete Nutzerprozess identifizieren. Die so ermittelte Prozess-ID kann nun an *Strace* angefügt und die Aufzeichnung gestartet werden. *Swingbench* erzeugt während dieses Vorgangs weiterhin Last auf dem System. Alle Transaktionen werden über den ermittelten Nutzerprozess geleitet. *Swingbench* wird weitere Fünfzehn Minuten fortgesetzt, bevor es beendet wird. Am Ende wird die erstellte Log-Datei auf geöffnete Bibliotheken hin untersucht. Der Ablauf diese Tests ist in Abbildung 5.2 nochmal dargestellt. Informationen zu den einzelnen Testdurchläufen sind dem Anhang B.1 auf Seite 81 zu entnehmen.

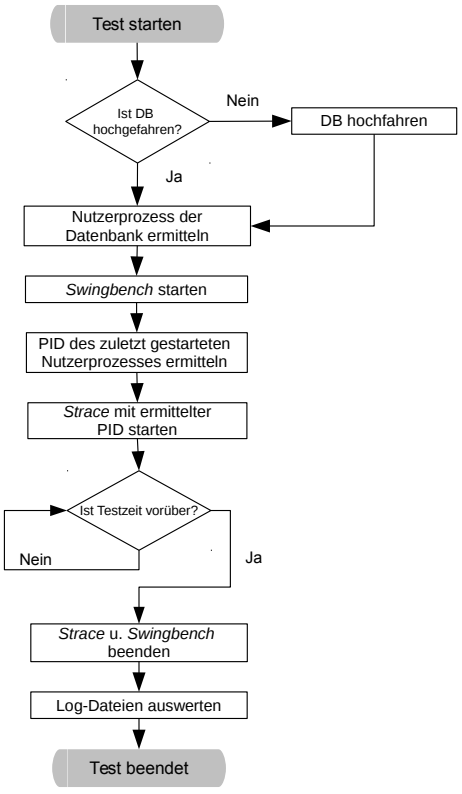

Abbildung 5.2: Ablaufdiagramm des Tests

5.3.2 Auswertung

Die mit *Strace* entstandenen Aufzeichnungen enthielten keine Einträge über Bibliotheken. Auch durch die Nutzung des Parameters `-f` änderte sich nichts am Ergebnis. Wohingegen das Abbild mittels *lsof* mehrere Bibliotheken auflistete, wie im Anlage A.3 zu sehen ist. Dieser Test hat gezeigt, dass es sehr aufwändig wäre alle Prozesse einzeln auf die Verwendung von Bibliotheken hin zu untersuchen, um diese später gezielt auszutauschen. Im Folgenden wird versucht mehrere Datenbankprozese gleichzeitig zu untersuchen und im Anschluss die entsprechenden Bibliotheken herauszufiltern. Tabelle 5.5 zeigt die erwarteten und tatsächlichen Ergebnisse dieser Untersuchung.

Erwartetes Ergebnis	**Tatsächliches Ergebnis**
Nutzerprozess lädt dynamisch Bibliotheken.	wurde nicht erfüllt
Nutzerprozess lädt gemeinsame Bibliotheken.	wurde erfüllt

Tabelle 5.5: Ergebnisse Nutzerprozess-Test

5.4 Voruntersuchung - Bibliotheken mehrerer Datenbankprozesse

Das Vorgehen bei der Analyse des Nutzerprozesses war nicht gut geeignet um Bibliotheken zu ermitteln, welche die Datenbank nutzt. Daher beobachtet diese Untersuchung möglichst viele Datenbankprozesse und filtert geeignete Beispiele für den nachfolgenden Test heraus.

Aus Gründen der Handhabung werden die Datenbankprozesse für die Tests in drei Gruppen unterteilt. Dies sind zum einen die Prozesse, die aus dem *Oracle*-Installationsordner (*Oracle-Home-Verzeichnis*) heraus gestartet werden, beispielsweise `/u01/app/12.1.0/grid_1/bin/evmd.bin`. Weiterhin sind das die Hintergrundprozesse der Datenbank (`ora_smon_orcl`) und als letzte Gruppe die Prozessen des Grid-Systems (`asm_pmon_+ASM`). Durch unterschiedliche Prozessgruppen erhofft sich der Autor zusätzliche Erkenntnisse über die Art und Verwendung der Bibliotheken.

5.4.1 Durchführung

Als erstes wurden mit dem Kommando `ps -ef | grep oracle` alle laufenden Prozesse der *Oracle*-Datenbank mit ihren Prozess-IDs ermittelt. Diese wurden Gruppiert und an *Strace* als Parameter angefügt. Anschließend wurde *Swingbench* gestartet. Für diese Untersuchung waren Fünfzehn Minuten ausreichend, um genügend Daten zu ermitteln. Die Aufzeichnungen wurden nach Bibliotheken durchsucht und daraus eine Auswahl an Beispielen erstellt.

5.4.2 Auswertung

Die Datenbankprozesse luden Bibliotheken aus zwei unterschiedlichen Quellen. Zum einen aus dem Installationsordner der Datenbank (*Oracle-Home-Verzeichnis*). Die darin enthaltenen Daten werden nur mit einem separaten Datenbank Update verändert und waren daher für diese Untersuchung uninteressant, z.B.: `/u01/app/oracle/product/11.2.0/dbhome_1/lib/libocrutl11.so`. Zum anderen wurden Daten aus den üblichen Systemordnern für Bibliotheken geladen, diese werden bei einem Systemupdate verändert, z.B.: `/lib64/libc-2.12.so`.

Über den Befehl `yum whatprovides [Bibliothek]` wurden die Pakete ermittelt, welche die angegebene Bibliothek beinhalteten. Diese sind in Tabelle 5.6 aufgelistet. Es ist zu erkennen, dass viele Bibliotheken aus dem Paket „glibc-2.12-1.7.el6.x86_64“ stammen.

Bibliothek	**Pakete**
/lib64/librt.so.1	glibc-2.12-1.7.el6.x86_64
/lib64/libaio.so.1	libaio-0.3.107-10.el6.x86_64
/lib64/libdl.so.2	glibc-2.12-1.7.el6.x86_64
/lib64/libm.so.6	glibc-2.12-1.7.el6.x86_64
/lib64/libc.so.6	glibc-2.12-1.7.el6.x86_64
/usr/lib64/libnuma.so.1	numactl-2.0.3-9.el6.x86_64

Tabelle 5.6: Zuordnung der Bibliotheken zu Paketen

Im Anschluss musste festgestellt werden, ob Aktualisierungen dieser Pakete existierten. Dies wird an dieser Stelle beispielhaft mit der Bibliothek `glibc` und dem Testsystem A verdeutlicht. Mit dem Kommando `yum list update glibc` wurde geprüft, ob eine aktuellere Version des Pakets vorlag. Dafür wurde in der Konfiguration des Paketmanagers das Repository (ol6_latest) angegeben, das die neuesten Pakete für die Distribution (*Oracle-Linux 6*) bereitstellt. In Tabelle 5.7 ist zu erkennen, dass es bei den Bibliotheken nur wenig Versionssprünge gibt und es sich bei den Aktualisierungen meist nur um Änderungen in der Release-Version handelt. Für das Paket *libaio* gibt es keine neue Version (Version / Release).

Version im System	**Version im Repository**	**Aktualisierungsgrad**
glibc-2.12-1.7.el6.x86_64	glibc-2.12-1.166.el6.7.1.x86_64	Release-Sprung
libaio-0.3.107-10.el6.x86_64	libaio-0.3.107-10.el6.x86_64	keine Updates
numactl-2.0.3-9.el6.x86_64	numactl-2.0.9-2.el6.x86_64	Versions-Sprung

Tabelle 5.7: Vergleich Paket-Versionen

Ein Paket enthält mehrere Dateien, darum ist es nicht eindeutig zu klären, ob die ermittelten Bibliotheken in den Paketen oder andere uninteressante Dateien aktualisiert wurden. Mit dem Programm *Midnight Commander* kann die Archivdatei im *RPM*-Paket betrachtet werden. Die zu untersuchenden Pakete wurden aus dem Repository (ol6_latest) heruntergeladen. Es konnte aber anhand des Dateinamens der Bibliothek keine Aktualisierung festgestellt werden. Die Pakete enthielten jedoch die zu untersuchenden Bibliotheken, d.h. diese würden bei einer Aktualisierung zumindest ausgetauscht werden. Für die folgenden Tests ist damit zu beachten, dass sich der Dateiname der Bibliothek nicht ändert, obwohl es sich um ein neues Release handelt.

Über eine andere Herangehensweise sollte nochmal geprüft werden, ob das Paket wirklich Aktualisierungen für die Bibliotheken bereitstellt. Dafür wurden die Quellcode-Pakete betrachtet und am Beispiel von „glibc-2.12-1.166.el6.src.rpm“ wie folgt analysiert. Die Archivdatei „CONTENTS.cpio“ wurde aus dem Paket heraus kopiert und entpackt. Mit den darin zu findenden Archivdateien ist analog verfahren worden. Eine Suche nach Bibliotheken, in der entpackten Verzeichnisstruktur ergab, dass für die Bibliotheken *librt*, *libm* und *libc* C-Dateien existierten. Lediglich für die Bibliothek *libdl* wurde nur die Datei „libdl.abilist“ ermittelt.

Der Test ergab, dass die Pakete „glibc.x86_64“ und „libaio.x86_64“, sowie die Bibliotheken *librt*, *libm* und *libc* von Interesse für die Durchführung des folgenden Tests sind.

5.5 Stabilitätstest mit Yum - Austauschbarkeit von Bibliotheken

Die *Oracle*-Datenbank startet und beendet Prozesse. dies gehört zur normalen Funktionsweise der Datenbank. Die Prozesse selber nutzen gemeinsame Bibliotheken, die beim Starten des Prozesses in den Arbeitsspeicher geladen werden. Wenn bei einem Update eine dieser Bibliotheken durch eine neuere Version ersetzt wird, müssen alle Prozesse, welche nach dem Update starten, die neue Bibliothek nutzen. Der Test soll klären, ob der Austausch von System-Bibliotheken während der Laufzeit einer *Oracle*-Datenbank möglich ist oder zu Problemen führt. Sollte dies funktionieren, würde anschließend die Stabilität des Systems und der Datenbank getestet.

5.5.1 Durchführung

Durch die Analyseprogramme *Strace* und *lsof* kann ermittelt werden, welche Bibliothek von einem Programm genutzt wird. Die interessante Frage ist, wann diese verwendet werden. Denn wenn eine der zu untersuchenden Bibliotheken sowohl vor als auch nach dem Systemupdate in den Arbeitsspeicher geladen wird, könnte es zu Problemen kommen.

Zu Beginn eines Tests wurde die Verwendung der Datenbank simuliert, indem *Swingbench* startete. Gleich danach erstellte *lsof* das erste Abbild der geöffneten Dateien. Mit diesem kann ermittelt werden, welche Bibliotheken in den Speicher geladen wurden. Anhand des zweiten Abbildes, was nach dem Systemupdate entstand, soll bestimmt werden, ob die verwendeten Bibliotheken von der Festplatte gelöscht wurden. Die Datenbank arbeitet mit zwei unterschiedlichen Bibliotheksversionen, wenn das System eine neue Version der Bibliothek in den Arbeitsspeicher lud. Ob dies Stabilitätsprobleme verursacht, wurde im Anschluss anhand eines mehrstündigen Testbetriebs erprobt. Während der gesamten Testphase protokolliert *Strace* die geöffneten Daten der Datenbankprozesse. Am Ende des Tests wurden die Log-Dateien ausgewertet.

Der Test wurde insgesamt vier mal durchgeführt, um ein reproduzierbares Ergebnis zu erhalten. Die Wiederholungen sollen nicht berücksichtigte Randbedingungen aufzeigen und die bisher festgestellten Ergebnisse bestätigen. Dabei wurde ein Testlauf auf dem Testsystem mit *Oracle-Linux 7* und die restlichen auf dem mit *Oracle-Linux 6* durchgeführt. Die Details jedes Testdurchlaufes sind dem Anhang C.1 zu entnehmen.

Das in Abbildung 5.3 zu sehende Ablaufdiagramm beinhaltet alle Vorgänge, die während eines Test durchgeführt werden können. Teilweise wurden einzelne Punkte ausgelassen, dies ist den jeweiligen Ablaufprotokollen (Anhang C.2) der Testdurchläufe zu entnehmen.

5.5.2 Auswertung

Alle Tests wurden auf die selbe Art ausgewertet. Folgend wird dies beispielhaft mit der Bibliothek *librt.so* aus dem Paket *glibc-2.12-1.7.el6.x86_64* getan. Die Datei *librt.so.1* zeigt lediglich auf die Datei *librt-2.12.so*. Anhand der Dateigröße und der Versionsnummer ist zu erkennen, dass *librt-2.12.so* den eigentlichen Binärcode der Bibliothek enthält. Das verdeutlicht der folgendem Konsolenausschnitt. Bei der Auswertung können beide Dateinamen gleichwertig genutzt werden.

```
[root@c-moh-ksplice-01 lib64]# ls -al | grep librt
-rwxr-xr-x 1 root root 47112 Sep 24 17:07 librt-2.12.so
lrwxrwxrwx 1 root root 13 Sep 25 17:10 librt.so.1 -> librt-2.12.so
```

Durch den Vergleich der Abbilder, welche mittels *lsof* aufgezeichnet wurden, ist zu erkennen, dass die Datenbankprozesse unterschiedliche Realeas-Versionen einer Bibliothek nutzen. Im folgenden Auszug ist zu sehen, dass die Bibliothek *librt-2.12.so* von dem Prozess *oracle* in den Arbeitsspeicher geladen wurde. Dies ist an der Abkürzung `mem` zu erkennen. [111] Dieser Auszug ist aus dem Abbild, welches vor dem Systemupdate durchgeführt wurde.

[111] *Vgl. Abell, Victor: lsof(8) - Linux man page.* [90]

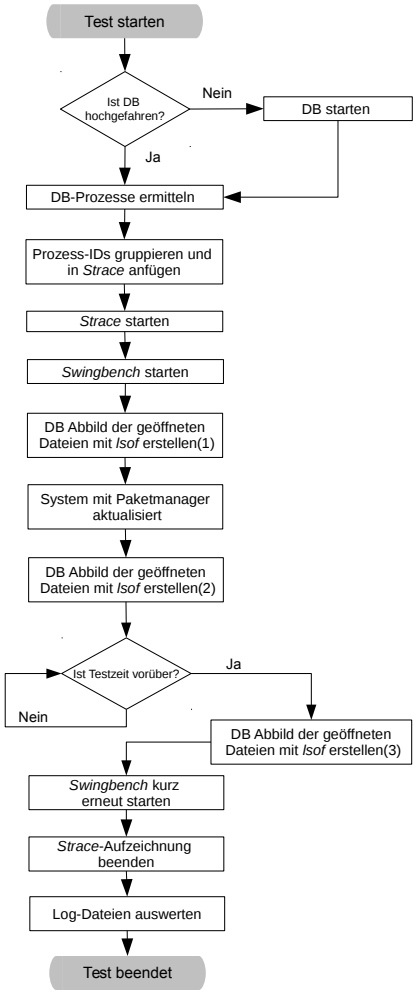

Abbildung 5.3: Ablaufdiagramm der Yum-Tests

```
oracle 4280 oracle mem REG 253,0 47072 786828 /lib64/librt-2.12.so
```

Bei dem Abbild, welches nach dem Systemupdate durchgeführt wurde, ist die Datei auf der Festplatte gelöscht wurden. Das ist an dem `DEL` zu erkennen. [111]

```
oracle 4280 oracle DEL REG 253,0 786828 /lib64/librt-2.12.so
```

Nach der Testdauer von Fünfzehn Stunden wurde die Bibliothek weiterhin als gelöscht angegeben, aber nur bei den Prozessen, welche sie vor Testbeginn geladen hatten. Prozesse die während des Tests starteten konnten die Bibliothek wieder von der Festplatte laden. Das ist gut an den PIDs und den Dateibeschreibungen `DEL` und `mem` zu erkennen.

```
oracle 4282 oracle DEL REG 253,0 786828 /lib64/librt-2.12.so
oracle 11948 oracle mem REG 253,0 47112 786656 /lib64/librt-2.12.so
```

Das gleiche Ergebnis konnte mit den Aufzeichnung erzielt werden, die mittels *Strace* durchgeführt wurden. Die Datenbankprozesse haben die Bibliothek *librt* vor und nach dem Systemupdate geladen. Dies ist am Zeitstempel zu sehen. Das Update wurde im

zweiten Testlauf 17:03 Uhr begonnen und 17:26 Uhr war das System fertig aktualisiert.

```
4256 16:57:09.127616 open(''/lib64/librt.so.1'', O_RDONLY) = 3
5938 17:15:03.931188 open(''/lib64/librt.so.1'', O_RDONLY) = 3
8930 18:21:29.588899 open(''/lib64/librt.so.1'', O_RDONLY) = 3
```

In allen Testdurchläufen wurden die gleichen Ergebnisse erzielt. Den Log-Dateien des System und der Datenbank konnten keine Einträge entnommen werden die auf eine Instabilität hindeuten.

Erwartetes Ergebnis	**Tatsächliches Ergebnis**
Datenbank läuft stabil (15 Stunden Testzeit)	wurde erfüllt
System läuft stabil (15 Stunden Testzeit)	wurde erfüllt
Datenbank nutzt unterschiedliche Releas-Versionen einer Bibliothek	wurde erfüllt
keine auffälligen Einträge in den Systemlogs	wurde erfüllt
keine auffälligen Einträge in den Datenbanklogs	wurde erfüllt

Tabelle 5.8: Ergebnisse *Yum*-Tests

Anhand der Tests ist zu erkennen, dass die Datenbankprozesse mit unterschiedlichen Release-Versionen von Bibliotheken stabil betrieben werden können. Dabei ist nochmal zu betonen, dass es sich bei den Versionsunterschieden wirklich nur im Ausnahmefall um wirklich neue Versionen handelt. In der Regel wurden nur kleine Veränderungen im Quellcode vorgenommen und es handelt sich lediglich um ein neues Release der Bibliothek (Minor-Release). Die Testsysteme funktionierten Fünfzehn Stunden lang mit unterschiedlichen Bibliotheksversionen stabil. Es konnten in den Aufzeichnungen keine Einträge gefunden werden, die auf eine Instabilität hinweisen.

5.6 Stabilitätstest mit Ksplice

Für *Ksplice* konnten keine Ansatzpunkte recherchiert werden, welche die Stabilität von *Oracle-Linux* gefährden könnten. Die Nachfolgenden Tests haben daher keinen Fokus auf eine spezielle Schwachstelle. Untersucht wird die Stabilität der Testsysteme A und B. Während des Betriebs einer *Oracle*-Datenbank werden Updates in den Linux-Kern des Systems eingespielt. Das *Oracle-Linux* soll dabei keine Instabilität aufweisen oder abstürzen. Die Testziele dieser Untersuchung lauten wie folgt.

1. Aktualisieren des Linux-Kerns mit *Ksplice*, ohne Systemfehler, Instabilität, Abstürze von Programmen oder des Testsystems.
2. Deinstallieren von *Ksplice*-Patches aus dem Kern, ohne Systemfehler, Instabilität, Abstürze von Programmen oder des Testsystems.

5.6.1 Durchführung

Bei der Durchführung der Tests wurde zwischen der *Oracle-Linux* Version 6 (Testsystem A) und 7 (Testsystem B) variiert. In beiden Systemen wurde der Kern zwischen den Testdurchläufen ausgetauscht, um zwischen einem *Unbreakable Enterprise Kernel* und einem *Red Hat Compatible Kernel* wechseln zu können. Dabei wurde die Installation und Deinstallation der *Ksplice*-Patches als einzelner Test aufgeführt. So entstanden acht Testdurchläufe, wie in Tabelle 5.9 zu sehen ist. Die Testprotokolle jedes einzelnen Tests sind im Anhang D.1 auf Seite 93 zu finden.

	Unbreakable Enterprise Kernel		*Red Hat Compatible Kernel*	
Oracle-Linux 7	Installation	Testlauf 1	Installation	Testlauf 3
	Deinstallation	Testlauf 2	Deinstallation	Testlauf 4
Oracle-Linux 6	Installation	Testlauf 5	Installation	Testlauf 7
	Deinstallation	Testlauf 6	Deinstallation	Testlauf 8

Tabelle 5.9: Übersicht der *Ksplice*-Testdurchläufe

Zu Beginn jedes Testdurchlaufes wurde *Swingbench* gestartet und abgewartet bis dieses eine durchschnittliche Transaktionsrate erreicht hatte, dies dauerte ca. 2 Minuten. Anschließend wurde mit dem Kommando `uptrack-upgrade` die Installation aller zur Verfügung stehenden *Ksplice*-Patches begonnen. Zum Entfernen aller installierten Patches wurde die Anweisung `uptrack-remove -all` verwendet, je nachdem welches Testszenario durchgeführt werden sollte. Wenn einer der Patches nicht installiert werden konnte, wurde der Test nicht sofort beendet, sondern einige Zeit gewartet. Dadurch sollte dem System die Möglichkeit gelassen werden die verwendeten Kern-Funktionen wieder freizugeben, wenn diese von Prozessen blockiert wurden. Die genauen Wartezeiten sind der Anlage auf Seite 93 zu entnehmen.

Wenn dies nicht funktionierte, wurden die blockierenden Prozesse mit `kill -9 PID` beendet. Anschließend wurde die Deinstallation bzw. Installation der Patches erneut gestartet. Wenn der Vorgang weiterhin bei dem selben Patch durch die gleichen Prozesse blockiert wurde, musste der Test an dieser Stelle beendet werden. Das Testsystem wurde in jedem Fall noch weitere Fünfzehn Minuten betrieben, um Fehler bei den bis dato durchgeführten Aktualisierungsvorgängen festzustellen. Ein längerer Test wäre gründlicher, aber ein Fehler lange nach dem Update-Vorgang würde eher auf einen fehlerhaften Patch zurückzuführen sein, als auf den Aktualisierungsvorgang. Im Anschluss jedes Tests wurden die Log-Dateien der Datenbank, von *Swingbench* und des Systems nach Einträgen untersucht. Abbildung 5.4 verdeutlicht den Ablauf der Testdurchläufe noch einmal.

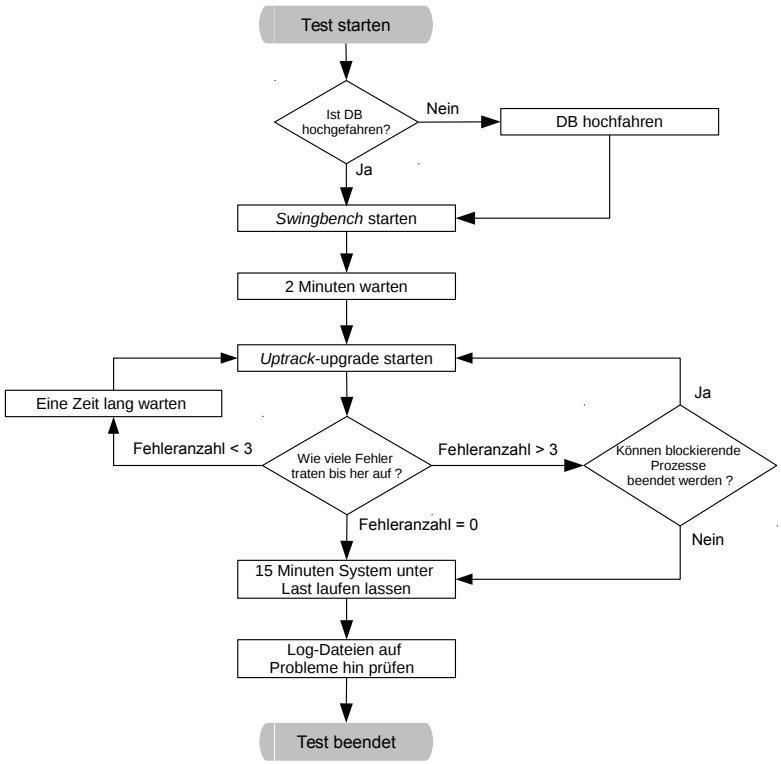

Abbildung 5.4: Ablaufdiagramm der *Ksplice*-Tests

5.6.2 Auswertung

Bei den acht durchgeführten Testdurchläufen gab es bei vier keine Unterbrechungen (Testdurchlauf 4,5,6,8). Zu beachten ist, dass im vierten Testdurchlauf lediglich vier Patches deinstalliert wurden, da es nicht möglich war alle Aktualisierungen im vorherigen Testdurchlauf zu installieren. Auch der Versuch die Patches beim Systemstart mit zu installieren funktionierte nicht. Dafür wurde in der *Uptrack* Konfiguration die Einstellungen `upgrade_on_reboot` einkommentiert und auf „yes“ gesetzt.

Die Testdurchläufe eins und sieben wurden bei einem Patch unterbrochen, da verschiedene Prozesse auf die Funktion zugegriffen hatten, welche ausgetauscht werden sollte. Es wurde mehrfach versucht die Installation zu einem späteren Zeitpunkt wieder fortzuführen. Dies gelang in beiden Test auch.

Lediglich in den Testdurchläufen zwei und drei konnten nicht alle Patches installiert bzw. deinstalliert werden. Auch das manuelle Beenden der Prozesse, welche den Patchvorgang blockierten, erbrachte keinen Erfolg. Denn diese wurden automatisch neu gestartet.

Im zweiten Testdurchlauf trat das Problem auch nach dem Abschalten von *Swingbench* und einer Wartezeit von 17 Stunden auf. Die Prozesse, welche die Deinstallation im zwei-

ten Testdurchlauf blockierten, hatten die Bezeichnung *kworker*. Sie gehören zu einem Prozess-Management-Konzept von Linux namens *Concurrency Managed Workqueues* (CMWQ). [112] Bei diesem Konzept wird, abhängig von der Anzahl der Prozessoren im System, ein Satz von Threads erstellt. Diese arbeiten Warteschlangen ab, sogenannte *Workqueues*. In denen befinden sich unterschiedliche Aufgaben (work) des Betriebssystems. Es ist recht schwierig zu analysieren, welche Aufgabe aus der Warteschlange gerade von einem der Threads abgearbeitet wird und welches Programm oder Dienst diese ausgelöst hat. [113] Das Beenden dieser Prozesse zu erzwingen endet in einem Neustart der selbigen unter anderen PIDs. Daher ist es schwierig das Deinstallieren der *Ksplice*-Patches zu erzwingen. Das Testsystem wurde drei weitere Tage ruhen gelassen, um zu prüfen, ob die entsprechenden Funktionen im Kern nicht mehr verwendet werden, ohne Erfolg.

Zu dem Problem wurde *Oracle* über ein Support-Ticket befragt, das dem Anhang E.3 auf Seite 116 zu entnehmen ist. *Oracle* empfahl einen Neustart des Systems, bei dem die automatische Installation der Patches während des Boot-Vorgangs ausgeschaltet bleibt. Dies erreicht man über den Parameter `install_on_reboot`, der in der Konfigurationsdatei von *Uptrack* gesetzt wird. Ein Live-Patching-System dient dazu einen Neustart zu vermeiden und keinen zu provozieren. Trotzdem wurde versucht diesen Fehler auf die empfohlene Art zu beheben, ohne Erfolg. Die Installation aller *Ksplice*-Patches und die erneute Deinstallation behoben das Problem auch nicht. Alle aufgetretenen Fehler sind im Anhang D.2 auf Seite 100 aufgelistet.

Die Testdurchläufe zeigen, dass das Aktualisieren mit *Ksplice* die Stabilität des Systems nicht gefährdet. Jedoch können blockierende Prozesse *Ksplice* über mehrere Tage lahm legen, da die Installation und Deinstallation nicht weiter fortgesetzt werden kann. Ein Neustart von Programmen oder des Betriebssystems ist an dieser Stelle schwierig zu vermeiden. Dies geht aus dem *Oracle*-Support-Protokoll vom 23.10.2015 hervor, siehe Anlage E.3. Die Tests haben aber auch gezeigt, dass dies nicht immer zum Erfolg führt. Die Prozesse können automatisch neu gestartet werden, bzw. beeinträchtigt deren Beendigung die Arbeit des Systems. Diese Vorgehensweise wurde nur aus Testgründen gewählt und ist in der Praxis nicht praktikabel.

Wie in Tabelle 5.10 zu sehen ist, wurden nicht alle erwarteten Ergebnisse erfüllt. In einigen Test konnten nicht alle Patches deinstalliert oder installiert werden. Probleme bei der Ausführung der Datenbank und des Systems wurden nicht festgestellt. Dabei ist zu beachten, dass diese Aussage nur direkt auf den Patchvorgang zutrifft. Das beheben von Verklemmungen kann Auswirkung auf die Stabilität des Systems, bzw. der darauf laufenden Programme haben. Außerdem bleibt der Kern inaktuell, wenn *Ksplice* nur vier Patches einspielt.

[112] *Vgl. Corbet, Jonathan: Concurrency-managed workqueues and thread priorities, 2010.* [91]
[113] *Vgl. Heo, Tejun / Mickler, Florian: Concurrency Managed Workqueue (cmwq), 2010.* [92]

Erwartetes Ergebnis	Tatsächliches Ergebnis
Datenbank läuft stabil	wurde erfüllt
System läuft stabil	wurde erfüllt
keine auffälligen Einträge in den Systemlogs	wurde erfüllt
keine auffälligen Einträge in den Datenbanklogs	wurde erfüllt
alle *Ksplice*-Patches wurden installiert	wurde nicht erfüllt
alle *Ksplice*-Patches wurden deinstalliert	wurde nicht erfüllt

Tabelle 5.10: Ergebnisse *Ksplice*-Tests

5.7 Fazit der Tests

Die Test mit *Ksplice* ergaben, dass dessen Verwendung nicht die Stabilität des Systems gefährdet. Es kann aber zu Verklemmungen mit Prozessen kommen, die sich auch nach mehreren Tagen nicht auflösen lassen. Da es nicht möglich ist einen *Ksplice*-Patch zu überspringen muss so lange gewartet werden bis sich die Verklemmung von alleine gelöst hat. Das kann dazu führen, dass der Linux-Kern für einen gewissen Zeitraum nicht aktuell ist. Die automatische Installation der Patches durch *Ksplice* kann das Problem abfedern aber nicht ganz beheben. Dazu konnte keine eigenen Untersuchungen durchgeführt werden, da der zeitlichen Rahmen nicht ausreichte.

Durch die Aktualisierungen von *Oracle-Linux* mit *Yum* wurden keine Probleme bei dem Betrieb der Datenbank oder des Betriebssystems festgestellt. Die verschiedenen Datenbankprozesse nutzten unterschiedliche Bibliotheksversionen. Das Laden einer neuen Version bei einem bereits gestarteten Prozesse konnte nicht festgestellt werden. Zusätzlich wurde festgestellt, dass die Datenbank versuchte zuerst die Bibliotheken aus dem *Oracle-Home-Verzeichnis* zu laden. Während der gesamten Tests kam es zu keinen instabilen Verhalten der Datenbank oder von *Oracle-Linux*, was auf die normale Nutzung von *Ksplice* zurückzuführen wäre.

6 Zusammenfassung und Ergebnis der Arbeit

Durch die Untersuchungen in dieser Arbeit ist der Autor zu der Auffassung gekommen, dass *Ksplice* und *Yum* keine geeigneten Werkzeuge sind, um Downtime auf *Oracle*-Datenbank-Servern zu reduzieren. Um die Antwort zu begründen wird die Fragestellung differenziert betrachtet. Dabei wird zuerst auf *Ksplice* eingegangen und eine allgemeine Einschätzung gegeben. Anschließend werden bestehende Probleme und mögliche Lösungen aufgezeigt. Durch eine Auflistung beispielhafter Einsatzszenarien, soll das Ergebnis besser in einen praktischen Kontext eingeordnet werden. Im darauf folgenden Unterkapitel wird auf *Yum* eingegangen. Hier ist das Ergebnis allgemeiner. Es muss nicht in unterschiedliche Kontexte gesetzt werden. Am Ende des Kapitels wird das Ergebnis der Arbeit zusammengefasst.

6.1 Auswertung zu Ksplice

Durch die Verwendung der Software wurde weder das Betriebssystem noch die Datenbank destabilisiert, was für den Einsatz von *Ksplice* spricht. Bei den praktischen Untersuchungen zeigten sich jedoch andere Schwächen des Programms. Mehrfach konnte der Aktualisierungsvorgang nicht beendet werden, da Prozesse bestimmte Funktionen des Kerns blockierten. In der längsten Untersuchen war das über mehrere Tage der Fall. Auf Grund des zeitlichen Rahmens der Arbeit war es nicht möglich zu überprüfen, wie lange es gedauert hätte bis die Blockierung von alleine beendet gewesen wäre. Solange die blockierenden Prozesse weiterhin auf die Funktionen des Kerns zugreifen, kann der Aktualisierungsvorgang nicht abgeschlossen werden, denn die einzelnen Patches hängen voneinander ab. Dies schmälert den Effekt von *Ksplice*. Es existiert eine Funktion zum automatischen Installieren von *Ksplice*-Patches. Mit dieser könnte sich der Schweregrad der Problematik verringern. Jedoch muss dazu noch eine Untersuchung durchgeführt werden, die in dieser Arbeit nicht mehr möglich war. Der momentane Untersuchungsstand kann die Verwendung von *Ksplice* jedoch nicht empfehlen.

Für die erwähnte Untersuchung müsste wie folgt vorgegangen werden. Die Testsysteme werden auf einen Snapshot zurückgesetzt, in dem noch keine Patches in die Kerne eingespielt sind. Über die *Uptrack*-Konfiguration wird die automatische Installation der Patches eingeschaltet, dies geschieht mit der Option `autoinstall = yes`. Über einen Zeitraum von mehreren Wochen werden in verschiedenen Zeitabständen alle eingespielten Patches geprüft. Dazu reicht jeweils eine kurzer Blick in das Webinterface. Wenn sich bei dieser Untersuchung heraus stellt, dass *Ksplice* die Patches in einem angemessenen

Zeitrahmen installiert, dann sollte diese Option favorisiert werden. Dann könnte auch die zentrale Fragestellung dieser Arbeit, im Sinne von *Ksplice*, bejaht werden.

Wenn ein Unternehmen trotzdem Überlegungen anstellt *Ksplice* einzusetzen, sollte es verschiedene Faktoren berücksichtigen. Dabei wäre zu nennen, dass *Ksplice* nicht nur auf einem, sondern auf vielen Linux Systemen eingesetzt werden kann, wenn es einmal lizensiert ist. Die Anschaffung wird lohnender, um so mehr Computersysteme damit ausgestattet werden.

Weitere Faktoren sind die Gefährdungsstufen der Systeme und die Sicherheitsmaßnahmen, die bereits getroffen wurden. *Ksplice* kann nur eine weitere Komponente eines umfassenderen Sicherheitskonzeptes sein. Die folgenden Szenarien sollen die Abschätzung der genannten Faktoren verdeutlichen und als Beispiele dienen.

1. Szenario: Kosten-Nutzen
   Für ein Unternehmen, dass nur einen einzelnen Server in einem gesicherten Intranet betreibt und regelmäßig (min. halbjährlich) Patchdays durchführt, lohnt sich der Einsatz von *Ksplice* weniger. Dabei kommt es auch auf die Größe des Unternehmens an und ob es den *Oracle Linux Premier* Status erst noch erwerben muss.
2. Szenario: Risiko
   Ein Datenbankserver, der nicht nur im Intranet, sondern direkt am Internet angebunden ist und damit einem größeren Risiko, durch Hacker oder Schadsoftware, ausgesetzt wird, sollte *Ksplice* einsetzen. Dies trifft nicht nur für Datenbankserver, sonder für alle Internet nahen Systeme zu, z.B.: Proxyserver, Firewallserver, Webserver.
3. Szenario: Support-Dienstleister (ASPICON)
   Die mit *Ksplice* verwalteten Computer können gruppiert werden. Dadurch hat ein Support-Dienstleister die Möglichkeit, ein oder mehrere Systeme zu einem Kunden zuzuordnen. *Ksplice* kann so konfiguriert werden, dass neu eingetragene Computersystemen zuerst nicht für den Empfang von Patches aus dem ULN berechtigt sind. Erst wenn der Kunde einen entsprechenden Supportvertrag mit dem Dienstleister hat, kann dieser *Ksplice*-Patches für den Kunden frei schalten. Diese Berechtigung kann auch über die *Ksplice*-API angesprochen und so in ein Firmeneigenes Verwaltungssystem des Dienstleisters, integriert werden. Dadurch lassen sich auch Automatismen realisieren, die den Verwaltungsaufwand minimieren.

Ksplice kann nur den Linux-Kern und nicht das gesamte Betriebssystem aktualisieren. Zudem haben die praktischen Untersuchungen gezeigt, dass es unter bestimmten Bedingungen seine Funktion nicht erfüllen kann. Daher lautet die Antwort für *Ksplice*: Nein, *Ksplice* ist kein geeignetes Werkzeug um Downtime auf einem Oracle-Datenbankserver zu reduzieren.

6.2 Auswertung zu Yum

Der Paketmanager *Yum* bringt keine Funktion zum Live-Patching mit. Trotzdem können Updates während des Datenbankbetriebes durchgeführt werden. Die Aktualisierung von Bibliotheken ist unproblematisch, da die neue Version von startenden Prozessen verwendet wird. Programme, welche die Bibliothek bereits vor der Aktualisierung verwendet haben, nutzen die alte Version bis zu einem Neustart weiter. Ein Administrator kann also nicht vermeiden, einzelne Dienste und Programme neu zu starten. Wenn sicher gestellt werden muss, dass alle Anwendungen die aktualisierten Versionen nutzen, ist ein Neustart des Systems das Praktikabelste. Bei einem Patchday verringert sich die Downtime, da die Aktualisierungen noch während der Betriebszeit eingespielt werden können. Die Konfigurationsdateien der einzelnen Anwendungen sollten nach einem Update kontrolliert werden. Je nach Einstellung kann *Yum* die bisherige Konfiguration überschreiben oder eine neue Version der Konfigurationsdatei ist zur alten Version dazu gespeichert worden.

Die Tests haben gezeigt, dass die Datenbank und Oracle-Linux keine Probleme mit der Aktualisierung von Systembibliotheken während des Betriebs haben. Die Problematik liegt eher darin, dass die Aktualisierungen erst dann verwendet werden, wenn die Programme neu gestartet, bzw. die neuen Bibliotheken in den Arbeitsspeicher geladen werden. Im Zweifelsfall kann nicht sicher gestellt werden, ob und welches Programm die Aktualisierungen nutzt und welches nicht. Die Anwendungen, welche kritisch sind, werden meist jene sein, die nicht neu gestartet werden. Daher lautet die Antwort für *Yum*: Nein, *Yum* ist kein geeignetes Werkzeug um Downtime auf einem *Oracle*-Datenbankserver zu reduzieren.

6.3 Fazit der Arbeit

Nach der theoretischen Recherche war das Ergebnis der Arbeit bereits abzuschätzen. Die praktischen Untersuchungen bestätigten es zusätzlich. *Ksplice* erhöht zwar den Schutz und die Stabilität des System, indem es das Live-Patching des Linux-Kerns ermöglicht. Downtimes können damit aber nur sehr geringfügig verringert werden, weil es sich nur auf den Kern des Betriebssystems beschränkt. Die aufgetretenen Verklemmungen zeigen außerdem eine größere Schwäche auf, welche die Wirksamkeit von *Ksplice* stark schmälert. Die Aktualisierungen mit *Yum* bedingen einen Neustart der Anwendung, bzw. des gesamten Betriebssystem. Die Updates können zwar eingespielt werden, aber sie sind erst nach einem Neustart der Anwendung wirksam. Je nach Programm oder Dienst ist das auf einem Server nur in Verbindung mit Downtime möglich. Außerdem kann keine Aussage darüber getroffen werden, welches Programm die aktualisierte Software nutzt und welches nicht. Daher lautet die abschließende Antwort auf die zentrale Frage dieser Arbeit: *Ksplice* und *Yum* sind keine geeigneten Werkzeuge, um Downtime auf *Oracle*-Datenbank-Servern zu reduzieren.

Literaturverzeichnis

[1] Sadeghi, Ahmad-Reza: Center for Advanced Security Research, Darmstadt 2014, http://www.cased.de/news/news/aktuelles/251
(Zugriff am: 2015-08-05).

[2] Intel Corporation (Hrsg.): Intel's Fundamental Advance in Transistor Design Extends Moore's Law, Computing Performance. Sixteen Eco-Friendly, Faster and 'Cooler' Chips Incorporate 45nm Hafnium-Based High-k Metal Gate Transistors, http://www.intel.com/pressroom/archive/releases/2007/20071111comp.htm
(Zugriff am: 2015-08-05).

[3] Shankland, Stephen: Linux development by the numbers: Big and getting bigger, 2013,
http://www.cnet.com/news/linux-development-by-the-numbers-big-and-getting-bigger/
(Zugriff am: 2015-08-05).

[4] Black Duck Software, Inc. (Hrsg.): MySQL,
https://www.openhub.net/p/mysql/analyses/latest/languages_summary
(Zugriff am: 2015-08-05).

[5] managedhosting.de GmbH (Hrsg.): Ausfallsicherheit von IT-Systemen, Version 2.0, S.2,
https://www.managedhosting.de/ _media/whitepaper_kosten_von_systemausfaellen_de.pdf
(Zugriff am: 2015-08-10).

[6] Bundesamt für Sicherheit in der Informationstechnik (Hrsg.): Band G, Kapitel 1: Einführung Hochverfügbarkeit eine herausfordernde Aufgabenstellung für ein professionelles IT-Service Management, 2013,
https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Hochverfuegbarkeit/BandG/G1_Einfuehrung.pdf?__blob=publicationFile
(Zugriff am: 2015-08-05).

[7] DataCore (Hrsg.): Aktuelle IT-Anforderungen im Gesundheitswesen, 2015,
http://whitepaper.computerwoche.de/uploads/files/9f5652065c646c2b5acc522a44cd3f494ff6ea95.pdf
(Zugriff am: 2015-08-28).

[8] Wiley-VCH Verlag GmbH & Co. KGaA (Hrsg.): Hochverfügbarkeit der IT-Infrastruktur wichtig für Patientensicherheit, 2007,
http://www.management-krankenhaus.de/topstories/it-kommunikation/hochverfuegbarkeit-der-it-infrastruktur-wichtig-fuer-patientensicherheit
(Zugriff am: 2015-08-28).

[9] CrunchBase (Hrsg.): Ksplice,
https://www.crunchbase.com/organization/ksplice
(Zugriff am: 2015-08-28).

[10] Thoma, Jörg: Oracle schließt Red Hat und Suse vom Support aus, 2011,
http://www.golem.de/1107/85158.html
(Zugriff am: 2015-08-05).

[11] ASPICON GmbH (Hrsg.):Firmenportrait, Chemnitz 2015,
https://www.aspicon.de/unternehmen/unternehmen/firmenportrait/
(Zugriff am: 2015-10-08).

[12] Hardt, Dennis: Werkzeuggestützte Softwareprüfung Statische Analyse und Metrik, S. 3, S. 6f, Hannover 2006,
http://www.se.uni-hannover.de/priv/lehre_2006sommer_seminar/Folien-Dennis_Hardt_Statische_Analyse_und_Metriken.pdf
(Zugriff am: 2015-08-06).

[13] Koschke, Rainer: Vorlesung Software Reengineering, S. 3ff, Bremen 2010,
https://www.informatik.uni-bremen.de/st/lehre/re10/dynamische_analyse.pdf
(Zugriff am: 2015-08-06).

[14] Krieglsteiner, Susann: Wissenschaft und Fachtheorie, Methoden und Techniken der Disziplin, S. 144,
https://www.me.hs-mittweida.de/studium/informationen-fuer-studenten-der-stamm-studiengaenge/abschlussarbeit.html
(Zugriff am: 2015-08-5).

[15] Brockhaus Enzyklopädie in 12 Bänden: Band 11 STIN-VERD, 21. Auflage, Update S. 609, Leipzig / Mannheim 2005.

[16] Ubuntu Deutschland e.V. (Hrsg.): Wiki, apt-get, 2015,
https://wiki.ubuntuusers.de/apt/apt-get?redirect=no.
(Zugriff am: 2015-10-13).

[17] Microsoft Corporation (Hrsg.): Service Pack und Update Center, 2015, http://windows.microsoft.com/de-de/windows/service-packs-download#sptabs=win 10 06.10.15. (Zugriff am: 2015-10-13).

[18] Bundesamt für Sicherheit in der Informationstechnik (Hrsg.): Maßnahmenkatalog, M 3.66 Grundbegriffe des Patch- und Änderungsmanagements, 2008, https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKataloge/Inhalt/_content/m/m03/m03066.html. (Zugriff am: 2015-10-25).

[19] Sneed, Harry u.a.: Software- Produktmanagement, Wartung und Weiterentwicklung bestehender Anwendungssysteme, 1. Auflage, Heidelberg 2005.

[20] Kernel Live Patching, in: Wikipedia, Die freie Enzyklopädie, 2015, https://de.wikipedia.org/w/index.php?title=Kernel_Live_Patching&oldid=142936896 (Zugriff am: 2015-12-11).

[21] Oechsel, Rainer: Verteilte Systeme und Entwicklung verteilter Anwendungen, in: Schneider, Uwe / Werner, Dieter (Hrsg.): Taschenbuch der Informatik, 6. Auflage, S. 419, München 2007.

[22] Schäfer, Sebastian: Schaeferblick Weblog, Push vs. Pull in der Informationsverteilung – eine Definitionsfrage?, 2009, https://schaeferblick.wordpress.com/2009/07/06/push-vs-pull-in-der-informationsverteilung-%E2%80%93-eine-definitionsfrage/ (Zugriff am: 2015-10-13).

[23] Brockhaus Enzyklopädie in 12 Bänden: Band FARE-STIM, 21. Auflage, Stabilität S. 618, Leipzig / Mannheim 2005.

[24] Oechsel, Rainer: Verteilte Systeme und Entwicklung verteilter Anwendungen, in: Schneider, Uwe / Werner, Dieter (Hrsg.): Taschenbuch der Informatik, 6. Auflage, S. 436, München 2007.

[25] Dettmer, Steffen / Hemm, Torsten: SelfLinux, Bibliotheken, Version 0.12.1, http://www-lehre.inf.uos.de/~ainf/2006/dokumentation/SelfLinux-0.12.1/html/bibliotheken04.html#d141e603 (Zugriff am: 2015-09-02).

[26] Wolf, Jürgen: Linux-Unix-Programmierung. Das umfassende Handbuch, 3.Auflage, Bonn 2009.

[27] Wladislaw, Eckhardt: Konzepte von Betriebssystem-Komponenten,Programmstart & dynamische Bibliotheken, 2005,
https://www4.cs.fau.de/Lehre/SS05/PS_KVBK/talks/Eckhardt_Handout.pdf
(Zugriff am: 2015-10-08).

[28] Glatz, Eduard: Betriebssysteme. Grundlagen, Konzepte, Systemprogrammierung, S.594 ff, Heidelberg 2006.

[29] Plötner, Johannes / Wendzel, Steffen: Einstieg in Linux. Linux verstehen und einsetzen, 5. Auflage, Bonn 2012.

[30] Wikibooks, Die freie Bibliothek. (Hrsg.): Linux-Praxisbuch: Syslog, 2015,
https://de.wikibooks.org/w/index.php?title=Linux-Praxisbuch:_Syslog&oldid=767892
(Zugriff am: 2015-10-08).

[31] Ubuntu Deutschland e.V. (Hrsg.): Wiki, Logdateien, 2015,
https://wiki.ubuntuusers.de/logdateien
(Zugriff am: 2015-10-08).

[32] Oracle Corporation (Hrsg.): Oracle Linux Security Guide for Release 6, Configuring and Using System Logging, 2015,
http://docs.oracle.com/cd/E37670_01/E36387/html/ol_log_sec.html
(Zugriff am: 2015-10-08).

[33] Stallman, Richard:GNU General Public License, Version 2, 1991
http://www.gnu.org/licenses/old-licenses/gpl-2.0.de.html
(Zugriff am: 2015-09-03).

[34] Siever, Ellen u.a.: LINUX in a Nutshell, 6. Auflage, Beijing u.a. 2009.

[35] Oracle Corporation (Hrsg.): Oracle Linux Support and Oracle VM Support Global Price List, 2014,
http://www.oracle.com/us/corporate/pricing/els-pricelist-070592.pdf
(Zugriff am: 2015-10-08).

[36] Oracle Corporation (Hrsg.):Oracle Data Sheet, Oracle Linux, 2014,
http://www.oracle.com/us/technologies/linux/oracle-linux-ds-1985973.pdf?ssSourceSiteId=ocomde
(Zugriff am: 2015-10-08).

[37] Oracle Corporation (Hrsg.): Oracle Linux Security Guide for Release 6, About the Unbreakable Linux Network, 2015,
http://docs.oracle.com/cd/E37670_01/E37355/html/ol_about_uln.html
(Zugriff am: 2015-10-08).

[38] Hunter, Jeffrey: DBA Tips Archive for Oracle, Install Oracle Database 11g R2 on Linux using Oracle ASM - (OL5), 2015,
http://www.idevelopment.info/data/Oracle/DBA_tips/Linux/LINUX_22.shtml
(Zugriff am: 2015-11-02).

[39] Solbach, Sebastian: Applikationsüberwachung mit 11gR2 Grid Infrastruktur - Am Beispiel der DBConsole,
http://www.oracle.com/webfolder/technetwork/de/community/dbadmin/tipps/grid_dbconsole/index.html
(Zugriff am: 2015-11-02).

[40] Oracle corp. (Hrsg.): Oracle Help Center - Database Concepts, Process Architecture, 2015,
http://docs.oracle.com/cd/E11882_01/server.112/e40540/process.htm#CNCPT008
(Zugriff am: 2015-11-02).

[41] Oracle corp. (Hrsg.): Oracle Help Center - Database Administrator's Guide, About Oracle Database Background Processes, 2015,
http://www.idevelopment.info/data/Oracle/DBA_tips/Linux/LINUX_22.shtml
(Zugriff am: 2015-11-02).

[42] Lee, Insup: DYMOS: A DYNAMIC MODIFICATION SYSTEM, 1983,
http://www.cis.upenn.edu/~lee/mydissertation.doc
(Zugriff am: 2015-11-02).

[43] University of Maryland (Hrsg.): DYNAMIC SOFTWARE UPDATING, 2011,
http://www.cs.umd.edu/projects/PL/dsu/software.shtml
(Zugriff am: 2015-11-02).

[44] Smith, Edward u.a.: KITSUNE, 2013,
http://kitsune-dsu.com/
(Zugriff am: 2015-11-02).

[45] Makris, Kristis: UpStare manual RELEASE_0-12-9, 2012,
http://files.mkgnu.net/files/upstare/doc/latest_manual/manual.pdf
(Zugriff am: 2015-11-02).

[46] The DynAMOS Team (Hrsg.): DynAMOS manual RELEASE_0-5-5, 2007,
http://files.mkgnu.net/files/dynamos/doc/latest_manual/manual.pdf
(Zugriff am: 2015-11-02).

[47] Chen, Haibo u.a.: POLUS: A POwerful live updating system, 2007,
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.167.3843
(Zugriff am: 2015-11-02).

[48] Ericsson Computer Science Laboratory (Hrsg.): ERLANG, GETTING STARTED, http://www.erlang.org/
(Zugriff am: 2015-11-02).

[49] Thoma, Jörg / Kißling, Kristian: Linux-Kernel 4.0 bringt Live-Patching, 2015, http://www.linux-magazin.de/NEWS/Linux-Kernel-4.0-bringt-Live-Patching
(Zugriff am: 2015-11-02).

[50] The Linux Kernel Organization, Inc. (Hrsg.): The Linux Kernel Archives, San Francisco 2015,
https://www.kernel.org/.
(Zugriff am: 2015-09-04).

[51] Open Source Lab (Hrsg.): Yum Package Manager, 2014,
http://yum.baseurl.org/wiki/releases
(Zugriff am: 2015-11-01).

[52] Fischer, Marcus: Der „neue“ Paketmanager DNF, 2015,
http://marcus-fischer.com/?p=99
(Zugriff am: 2015-11-01).

[53] Heilig, Tobias: OneGet – der neue Paketmanager unter Windows 10, 2015,
https://www.escde.net/oneget-der-neue-paketmanager-unter-windows-10-2/
(Zugriff am: 2015-11-01).

[54] Oracle Corporation (Hrsg.): Oracle Buys Ksplice. Oracle Linux Enhanced with Zero Downtime Software Updates, Redwood Shores Calif. 2011,
http://www.oracle.com/us/corporate/press/435791
(Zugriff am: 2015-08-11).

[55] Coekaerts, Wim / Oracle Corporation (Hrsg.): Using Oracle Ksplice to Update Oracle Linux Systems Without Rebooting, Revision 1.0, 2011,
http://www.oracle.com/technetwork/articles/servers-storage-admin/ksplice-linux-518455.html
(Zugriff am: 2015-08-11).

[56] Oracle Corporation (Hrsg.): Customer Support,
https://www.ksplice.com/legacy#supported-kernels
(Zugriff am: 2015-08-11).

[57] Oracle Corporation (Hrsg.): Ksplice Desktop,
http://ksplice.oracle.com/try/desktop
(Zugriff am: 2015-08-11).

[58] Oracle Corporation (Hrsg.): Uptrack, Installing Uptrack,
http://www.ksplice.com/uptrack/install
(Zugriff am: 2015-08-11).

[59] Oracle Corporation (Hrsg.): Oracle Linux Ksplice User's Guide, Registering to Use Oracle Ksplice, 2015,
http://docs.oracle.com/cd/E37670_01/E39380/html/ol_register_ksplice.html
(Zugriff am: 2015-08-11).

[60] Oracle Corporation (Hrsg.): Features,
http://ksplice.oracle.com/technology
(Zugriff am: 2015-08-11).

[61] Oracle Corporation (Hrsg.):Ksplice Uptrack API,
http://www.ksplice.com/uptrack/api
(Zugriff am: 2015-08-11).

[62] Oracle Corporation (Hrsg.): Ksplice User's Guide. 2.1 About the Ksplice Uptrack API, 2015
http://docs.oracle.com/cd/E37670_01/E39380/html/ol_about_kspapi.html
(Zugriff am: 2015-08-11).

[63] Oracle Corporation (Hrsg.): Uptrack. Uptrack Web Interface,
http://www.ksplice.com/uptrack/web
(Zugriff am: 2015-08-11).

[64] Oracle Corporation (Hrsg.):Ksplice Offline Client, Overview,
http://www.ksplice.com/uptrack/offline
(Zugriff am: 2015-08-11).

[65] Oracle Corporation (Hrsg.): Uptrack User's Guide,
http://www.ksplice.com/uptrack/guide
(Zugriff am: 2015-08-11).

[66] Arnold, Jeff / Kaashoek Frans: Ksplice: Automatic Rebootless Kernel Updates, Nuremberg 2009,
http://pdos.csail.mit.edu/papers/ksplice:eurosys.pdf
(Zugriff am: 2015-08-17).

[67] Streicher, Martin: Speaking UNIX: Get to know Ksplice. Give reboots the boot, S. 3, 2010,
http://www.ibm.com/developerworks/aix/library/au-spunix_ksplice/au-spunix_ksplice-pdf.pdf
(Zugriff am: 2015-08-18).

[68] Free Software Foundation, Inc.(Hrsg.): 3.10 Options That Control Optimization, 2015,
https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html#Optimize-Options
(Zugriff am: 2015-08-18).

[69] Turmbull, James u.a.: Pro Linux System Administration, Berkeley 2009.

[70] Brown, Robert / Pickard, Jonathan: Yum (Yellowdog Updater, Modified) HOWTO, 2003,
https://www.phy.duke.edu/~rgb/General/yum_HOWTO/yum_HOWTO/yum_HOWTO-1.html
(Zugriff am: 2015-08-19).

[71] Lauridsen, Tim : Yum Extender, About, 2014,
http://www.yumex.dk/p/about.html
(Zugriff am: 2015-08-24).

[72] OMcCallum, Ethan: Managing RPM-Based Systems with Kickstart and Yum, Sebastopol Calif. 2007,
http://proquest.tech.safaribooksonline.de/9780596513825
(Zugriff am: 2015-08-20).

[73] Oracle Corporation (Hrsg.): Oracle Linux Security Guide for Release 6, 2.2 Yum Configuration, 2015,
https://docs.oracle.com/cd/E37670_01/E37355/html/ol_yum_config.html
(Zugriff am: 2015-11-18).

[74] Bailey, Edward u.a.: Maximum RPM. Taking the Red Hat Package Manager to the Limit, Version 1.2, 2000,
http://www.rpm.org/max-rpm/index.html
(Zugriff am: 2015-08-25).

[75] Merker, Karsten: RPM-Pakete im Eigenbau, 1999,
https://www.unix-ag.uni-kl.de/~linux/linuxtag99/rpm_pakete_im_eigenbau.html
(Zugriff am: 2015-08-25).

[76] Welsh, Matt u.a.: Linux Wegweiser zur Installation & Konfiguration, 3. Auflage, 2000,
http://www.oreilly.de/german/freebooks/rlinux3ger/ch074.html
(Zugriff am: 2015-08-24).

[77] Kalhammer, Florian: Verwendung des Red Hat Package Managers (RPM), Version 1.102.6, 2002,
http://www.linux-praxis.de/lpic1/lpi101/1.102.6.html
(Zugriff am: 2015-08-24).

[78] Hilzinger, Marcel: Effizient Pakete verteilen mit Delta-RPM und Delta-ISO, in Linux Magazin vom 09.2005,
http://www.linux-magazin.de/Ausgaben/2005/09/Aussen-RPM-innen-Delta
(Zugriff am: 2015-08-25).

[79] Kofler, Michael : Linux 2011, Debian, Fedora, openSUSE, Ubuntu, 10. Auflage ,München u.a. 2011.

[80] Guillou, Gregory: Fast is the new better, Speeding Up Oracle Linux 7 Updates with „Delta RPMs“, 2014,
http://www.resetlogs.com/2014/08/ol7-deltarpms.html
(Zugriff am: 2015-08-25).

[81] Birnthaler, Thomas / Gottschalk, Hermann: HOWTO zu RPM (RedHat Package Manager), Version 1.14, 2013,
https://www.ostc.de/howtos/unix-rpm-HOWTO.html
(Zugriff am: 2015-09-02).

[82] Novell, Inc. (Hrsg.): openSUSE-Dokumentation, 2007,
http://www.mpipks-dresden.mpg.de/~mueller/docs/suse10.3/opensuse-manual_de/manual/index.html
(Zugriff am: 2015-09-02).

[83] Warbrick, Jon: RPM, %config, and (noreplace),
http://www-uxsup.csx.cam.ac.uk/~jw35/docs/rpm_config.html
(Zugriff am: 2015-11-15).

[84] Bailey, Edward u.a.: Maximum RPM. Taking the Red Hat Package Manager to the Limit, Using RPM to Upgrade Packages, Version 1.2, 2000,
http://www.rpm.org/max-rpm/ch-rpm-upgrade.html
(Zugriff am: 2015-11-15).

[85] Steffen Dettmer: Self Linux. Bibliotheken, Version 0.12.3,
http://www.selflinux.org/selflinux/html/bibliotheken03.html
(Zugriff am: 2015-09-02).

[86] Richter, Adam u.a.: dlopen(3) - Linux man page, Release 4.02, 2015,
http://man7.org/linux/man-pages/man3/dlopen.3.html
(Zugriff am: 2015-10-05).

[87] Hochstätter, Christoph: Enterprise-Linux: Red-Hat-Alternativen im Vergleich, 2012,
http://www.zdnet.de/41559153/enterprise-linux-red-hat-alternativen-im-vergleich/
(Zugriff am: 2015-09-02).

[88] Giles, Dominic: dominicgiles.com, 2010,
http://dominicgiles.com/index.html
(Zugriff am: 2015-10-12).

[89] Giles, Dominic: SwingBench, Reference and User Guide, 2002,
http://dominicgiles.com/swingbench/swingbench21f.pdf
(Zugriff am: 2015-10-12).

[90] Abell, Victor: lsof(8) - Linux man page,
http://linux.die.net/man/8/lsof
(Zugriff am: 2015-10-01).

[91] Corbet, Jonathan: Concurrency-managed workqueues and thread priorities, 2010,
http://lwn.net/Articles/393171/
(Zugriff am: 2015-11-02).

[92] Heo, Tejun / Mickler, Florian: Concurrency Managed Workqueue (cmwq), 2010,
http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/tree/Documentation/ workqueue.txt?id=HEAD
(Zugriff am: 2015-11-02).

[93] Kalhammer, Florian: Das Systemlogbuch, 2001,
http://www.linux-praxis.de/linux1/syslog.html
(Zugriff am: 2015-10-08).

[94] Hollowell, Christopher u.a.: Rebootless Linux Kernel Patching with Ksplice Uptrack at BNL, in : Journal of Physics: Conference Series, Volumen 396 Part 4, 2012,
http://iopscience.iop.org/1742-6596/396/4/042028/
(Zugriff am: 2015-08-05).

[95] Bailey, Edward: Maximum RPM. The ultimate source for advanced RPM packing technique, Indiana 1997.

Anhang A: Basistest

A.1 Testdurchläufe

Testlauf Nr.:	**1**
Konfiguration:	
Testzeit	17.09.15 17:00 - 18.09.15 08:09 Uhr
Kern	2.6.32-71.el6.x86_64
Oracle	*Oracle-Linux 6*
Swingbench	10 Nutzer, standard Transaktionen
Ergebnisse:	
Auffälligkeiten	keine
program.log	keine Aufzeichnung
dienste.log	keine Aufzeichnung
kern.log	keine Aufzeichnung
alert.log	kein neuer Eintrag
Swingbench	Keine Auffälligkeiten in den Logs
Datenbank / Grid	*Swingbench* startete normal, DB verarbeitete Transaktionen
Oracle-Linux	System reagiert angemessen, keine Probleme festzustellen

Testlauf Nr.:	**2**
Konfiguration:	
Testzeit	21.10.15, 16:35 - 22.09.15 08:40 Uhr
Kern	3.8.13-35.3.1.el7uek.x86_64
Oracle	*Oracle-Linux 7*
Swingbench	10 Nutzer, standard Transaktionen
Abweichungen	keine
Ergebnisse:	
Auffälligkeiten	keine
program.log	keine Aufzeichnung
dienste.log	keine Aufzeichnung
kern.log	keine Aufzeichnung
alert.log	keine Aufzeichnung
Swingbench	Keine Auffälligkeiten in den Logs,
Datenbank / Grid	*Swingbench* startete normal, DB verarbeitete Transaktionen
Oracle-Linux	System reagiert angemessen, keine Probleme festzustellen

Tabelle A.1: Programm-Parameter-Übersicht

A.2 Ablaufprotokoll

Tätigkeit **Kommando**	**Zeit**
Prozess-Id der laufenden Datenbankprozesse ermitteln. ps -ef | grep oracle	16:45 Uhr
Strace für Hintergrund-, *Oracle*-Home- und Grid-Prozesse starten. strace -f -tt -o straceOracleHomeProLTT.log -p 1937 ...	17:01 Uhr
Den Lastgenerator *Swingbench* starten. Startbutton drücken	17:05 Uhr
Abbilder der offenen Dateien aller *Oracle*-Prozesse erstellen. lsof -c oracle | grep .so >lsofVergleichstest.log	17:08 Uhr
System auf Funktionsfähigkeit prüfen.	08:09 Uhr 18.09.15

Tabelle A.2: Ablaufprotokoll

A.3 Geladene Bibliotheken des Nutzerprozesses

```
oracle 1738 oracle mem REG 253,0 1926520 787061 /lib64/libc-2.12.so
oracle 1738 oracle mem REG 253,0 22536 787062 /lib64/libdl-2.12.so
oracle 1738 oracle mem REG 253,0 145896 787073 /lib64/libpthread-2.12.so
oracle 1738 oracle mem REG 253,0 599392 787064 /lib64/libm-2.12.so
oracle 1738 oracle mem REG 253,0 47112 787077 /lib64/librt-2.12.so
oracle 1738 oracle mem REG 253,0 116368 786847 /lib64/libnsl-2.12.so
oracle 1738 oracle mem REG 253,0 5624 786660 /lib64/libaio.so.1.0.1
oracle 1738 oracle mem REG 253,0 65928 786461 /lib64/libnss_files-2.12.so
```

Anhang B: Voruntersuchung

B.1 Testdurchläufe des Nutzerprozess

Test Nr.:	**1**
Abweichungen	keine, wie oben beschrieben
Auffälligkeiten	Es wurden mehrere statt einem Nutzerprozess erzeugt
Ergebnis	Keine dynamischen Bibliotheken geöffnet, durchschnittlich 974 Transaktionen pro Minute.

Test Nr.:	**2**
Abweichungen	*Strace* mit zusätzlichem Parameter: -f
Auffälligkeiten	Es wurden mehrere statt einem Nutzerprozess erzeugt
Ergebnis	Keine dynamischen Bibliotheken geöffnet, durchschnittlich 954 Transaktionen pro Minute.

Test Nr.:	**3**
Abweichungen	*Strace* mit zusätzlichem Parameter: -f
Auffälligkeiten	Es wurden mehrere statt einem Nutzerprozess erzeugt
Ergebnis	Keine dynamischen Bibliotheken geöffnet, durchschnittlich 969 Transaktionen pro Minute.

Test Nr.:	**4**
Abweichungen	Testzeit auf 3 Stunden erhöht. *Strace* mit `-e open` Parameter. Mit *lsof* wurden ein Abbild der geöffneten Bibliotheken erstellt.
Auffälligkeiten	Es wurden mehrere statt einem Nutzerprozess erzeugt
Ergebnis	Es wurde eine Bibliothek dynamisch geladen, diese gehörte aber nicht zum System: /opt/oracle/extapi/64/asm/orcl/1/libasm.so

Tabelle B.1: Programm-Parameter-Übersicht

Anhang C: Stabilitätstest Yum

C.1 Testdurchläufe

Testlauf Nr.:	**1**
Konfiguration:	
Testzeit	von 21.09.15 16:43 Uhr bis 22.9.15 08:20 Uhr
Kern	2.6.32-71.el6.x86_64 / RCK
Oracle	*Oracle-Linux 6*
Swingbench	10 Nutzer, standard Transaktionen
Strace	-f -tt -e open -o -p
lsof	lsof -c oracle | grep .so > DATEIPFAD
Bemerkung:	
	Es gibt keine Übereinstimmung bei den Prozess IDs in den Aufzeichnungen von *Strace* und *lsof*.

Testlauf Nr.:	**2**
Konfiguration:	
Testzeit	von 24.09.15 16:35 Uhr bis 25.09.15 08:47 Uhr
Kern	2.6.32-71.el6.x86_64 / RCK
Oracle	*Oracle-Linux 6*
Swingbench	10 Nutzer, standard Transaktionen
Strace	-f -tt -o -p
lsof	lsof -c oracle | grep .so > DATEIPFAD
Bemerkung:	
	Strace konnte ab 21:00 Uhr die Log-Datei nicht mehr beschreiben, da der Speicher des Testsystems voll war! Tatsächlich aufgezeichnete Zeit: 4 Stunden.

Testlauf Nr.:	**3**
Konfiguration:	
Testzeit	von 30.09.15 15:50 Uhr bis 01.10.15 08:15 Uhr
Kern	2.6.32-71.el6.x86_64 / RCK
Oracle	*Oracle-Linux 6*
Swingbench	10 Nutzer, standard Transaktionen
Strace	-f -e open -o -p
lsof	lsof -c oracle | grep .so > DATEIPFAD

Bemerkung:	Fehler beim Aufzeichnen der *Oracle*-Home Prozesse Test wurde trotzdem gewertet, da er eine Aussage über die Stabilität zulässt.

Testlauf Nr.:	**4**
Konfiguration:	
Testzeit	von 29.10.15 16:20 Uhr bis 30.10.15 08:05 Uhr
Kern	3.10.0-123.el7.x86_64 / UEK
Oracle	*Oracle-Linux 7*
Swingbench	15 Nutzer, standard Transaktionen
Strace	-f -tt -e open -o -p
lsof	lsof -c oracle | grep .so > DATEIPFAD
Bemerkung:	

Tabelle C.1: Programm-Parameter-Übersicht

C.2 Ablaufprotokolle

Ablaufprotokoll Durchlauf 1:

Tätigkeit **Kommando**	**Zeit**
Prozess-Id der noch laufenden Datenbankprozesse ermitteln. ps -ef | grep oracle	16:42
Strace für Hintergrund-, *Oracle*-Home und Grid- Prozesse starten strace -f -o straceOracleHomeProLTT.log -p 1937 ...	16:43
Swingbench starten Startbutton	16:43
Abbilder der offenen Dateien aller *Oracle*-Prozesse (1) lsof -c oracle | grep .so >lsofOffeneLibsNachStart.log	16:45
System mit Paketmanager aktualisiert yum -y update (von ol6_latest)	17:03
Abbilder der offenen Dateien aller *Oracle*-Prozesse (2) lsof -c oracle | grep .so >lsofOffeneLibsNachTest.log	08:15 22.09.15
System auf Funktionsfähigkeit prüfen Log-Dateien Prüfen, *Swingbench* starten	08:20 22.09.15

Tabelle C.2: Ablaufprotokoll Testdurchlauf 1

Ablaufprotokoll Durchlauf 2:

Tätigkeit **Kommando**	**Zeit**
Datenbank in SQL*Plus herunterfahrenen shutdown immediate	16:35
Prozess-Id der noch laufenden Datenbankprozesse ermitteln. ps -ef | grep oracle	16:36
Strace für Hintergrund-, *Oracle*-Home und Grid- Prozesse starten strace -f -o straceOracleHomeProLTT.log -p 1937 ...	16:57
Datenbank hochfahren startup	16:57
Swingbench starten Startbutton	17:01
Abbilder der offenen Dateien aller *Oracle*-Prozesse (1) lsof -c oracle | grep .so >lsofOffeneLibsNachStart.log	17:01
System mit Paketmanager aktualisiert yum -y update (von ol6_latest)	17:03
Abbilder der offenen Dateien aller *Oracle*-Prozesse (2) lsof -c oracle | grep .so >lsofOffeneLibsNachUpdate.log	17:26
Abbilder der offenen Dateien aller *Oracle*-Prozesse (3) lsof -c oracle | grep .so >lsofOffeneLibsNachTest.log	08:47 25.09.15

Tabelle C.3: Ablaufprotokoll Testdurchlauf 2

Ablaufprotokoll Durchlauf 3:

Tätigkeit **Kommando**	**Zeit**
Prozess-Id der laufenden Datenbankprozesse ermitteln. ps -ef | grep oracle	15:50
Strace für Hintergrund-, *Oracle*-Home und Grid- Prozesse starten strace -f -o straceOracleHomeProLTT.log -p 1937 ...	16:03
Swingbench starten Startbutton	16:03
Abbilder der offenen Dateien aller *Oracle*-Prozesse (1)	

lsof -c oracle | grep .so >lsofOffeneLibsNachStart.log	16:06
System mit Paketmanager aktualisiert yum -y update (von ol6_latest)	16:40
Abbilder der offenen Dateien aller *Oracle*-Prozesse (2) lsof -c oracle | grep .so >lsofOffeneLibsNachUpdate.log	16:41
Abbilder der offenen Dateien aller *Oracle*-Prozesse (3) lsof -c oracle | grep .so >lsofOffeneLibsNachTest.log	08:15

Tabelle C.4: Ablaufprotokoll-Testdurchlauf3

Ablaufprotokoll Durchlauf 4:

Tätigkeit **Kommando**	**Zeit**
Prozess-Id der laufenden Datenbankprozesse ermitteln. ps -ef | grep oracle	16:20
Strace für Hintergrund-, *Oracle*-Home und Grid- Prozesse starten strace -f -o straceOracleHomeProLTT.log -p 1937 ...	16:23
Swingbench starten Startbutton	16:24
Abbilder der offenen Dateien aller *Oracle*-Prozesse (1) lsof -c oracle | grep .so >lsofOffeneLibsNachStart.log	16:25
System mit Paketmanager aktualisiert yum -y update (von ol6_latest)	16:40
Abbilder der offenen Dateien aller *Oracle*-Prozesse (2) lsof -c oracle | grep .so >lsofOffeneLibsNachUpdate.log	
Abbilder der offenen Dateien aller *Oracle*-Prozesse (3) lsof -c oracle | grep .so >lsofOffeneLibsNachTest.log	08:05

Tabelle C.5: Ablaufprotokoll-Testdurchlauf4

Anhang D: Stabilitätstest Ksplice

D.1 Testdurchläufe

Testlauf Nr.:	**1**
Konfiguration:	
Testzeit	22.10.15, 15:55-16:40 Uhr, 45 Minuten
Ksplice-Version	1.2.24
Kern	3.8.13-35.3.1.el7uek.x86_64 / UEK
Oracle	*Oracle-Linux 7*
Swingbench	15 Nutzer, standard Transaktionen
Abweichungen	keine, wie oben beschrieben
Ergebnisse:	
Auffälligkeiten	Instrallation von Patch [vfi7yfgq] schulg fehl.
program.log	keine Aufzeichnung
dienste.log	keine Aufzeichnung
kern.log	keine Aufzeichnung
alert_orcl.log	kein ungewöhlicher Eintrag
alert_+ASM.log	kein neuer Eintrag
alert.log	kein neuer Eintrag
Swingbench	Keine Auffälligkeiten in den Logs, 14479 TPM
Datenbank / Grid	*Swingbench* startete normal, DB verarbeitete Transaktionen
Oracle-Linux	System reagiert angemessen, keine Probleme festzustellen
Auswertung:	
	Die Installation aller *Ksplice*-Patches wurde bei dem Patch [vfi7yfgq] unterbrochen, da verschiedene Prozesse auf die Funktion zugegriffen hatten, welche ausgetauscht werden sollte. Nach drei erneuten Patchversuchen wurde der entsprechende Programmcode nicht mehr verwendet und konnte aktualisiert werden. Die Wartezeit zwischen den Versuchen betrug nur ca. fünf Minuten.

Testlauf Nr.:	**2**
Konfiguration:	
Testzeit	22.10.15, 16:50-17:15 Uhr, 25 Minuten Last + 17 Stunden Wartezeit
Ksplice-Version	1.2.24
Kern	3.8.13-98.2.1.el7uek.x86_64 / UEK
Oracle	*Oracle-Linux 7*
Swingbench	15 Nutzer, standard Transaktionen

Abweichungen	Deinstallation aller *Ksplice*-Patches

Ergebnisse:

Auffälligkeiten	Deinstallation von Patch [bejk62l4] schlug fehl.
program.log	keine Aufzeichnung
dienste.log	keine Aufzeichnung
kern.log	keine Aufzeichnung
alert_orcl.log	kein ungewöhlicher Eintrag
alert_+ASM.log	kein ungewöhlicher Eintrag
alert.log	keine Aufzeichnung
Swingbench	Keine Auffälligkeiten in den Logs, 13937 TPM
Datenbank / Grid	*Swingbench* startete normal, DB verarbeitete Transaktionen
Oracle-Linux	System reagiert angemessen, keine Probleme festzustellen

Auswertung:

Die Deinstallation aller *Ksplice*-Patches wurde bei dem Patch [bejk62l4] unterbrochen. Der Fehler lautete: `mis-handling of int80 fork from 64bits application.` Die Deinstallation wurde noch weitere drei mal gestartet, zwischen ihnen lag jeweils eine Wartezeit von ca. 5 Minuten. Das manuelle Beenden der Prozesse, die die Deinstallation blockierten, erbrachte keinen Erfolg. Denn diese werden automatisch neu gestartet. Nach abschalten von *Swingbench* und einer weiteren Wartezeit von 17 Stunden trat der Fehler weiterhin auf. Das Beenden dieser Prozesse zu erzwingen endet in einem Neustart der selbigen unter anderen PIDs. Daher war es schwierig das Deinstallieren der *Ksplice*-Patches zu erzwingen. Das Testsystem wurde drei Tage ruhen gelassen, um zu prüfen, ob die entsprechenden Funktionen im Kern nicht mehr verwendet werden, ohne Erfolg. Die Installation aller *Ksplice*-Patches und die erneute Deinstallation behoben das Problem auch nicht.

Testlauf Nr.:	**3**

Konfiguration:

Testzeit	28.10.2015, 11:45-12:45 Uhr, 60 Minuten
Ksplice-Version	1.2.24
Kern	3.10.0-123.el7 / RCK
Oracle	*Oracle-Linux 7*
Swingbench	15 Nutzer, standard Transaktionen
Abweichungen	

Ergebnisse:

Auffälligkeiten	Installation aller Patches scheiterte an Patch [aafjnrui]
program.log	keine Aufzeichnung
dienste.log	keine Aufzeichnung
kern.log	keine Aufzeichnung
alert_orcl.log	kein ungewöhlicher Eintrag
alert_+ASM.log	kein ungewöhlicher Eintrag
alert.log	keine Aufzeichnung
Swingbench	Keine Auffälligkeiten in den Logs. 11067 TPM
Datenbank / Grid	*Swingbench* startete normal, DB verarbeitete Transaktionen
Oracle-Linux	System reagiert angemessen, keine Probleme festzustellen

Auswertung:

Die Installation aller *Ksplice*-Patches (72) wurde durch den Fehler `Install [aafjnrui] CVE-2014-3153: Local privilege escalation in futex requeueing` unterbrochen. Der Vorgang wurde mehrfach wiederholt. Die längste Wartezeit zwischen den Versuchen betrug ca. 45 Minuten. Lediglich 4 von 72 *Ksplice*-Patches konnten eingespielt werden, da die Aktualisierung immer an dem Patch `[aafjnrui]` scheiterte. Das Beenden dieser Prozesse brachte die Datenbank zum einstürzen. Eine Installation des Patches war im Anschluss trotzdem nicht möglich!

Testlauf Nr.: 4

Konfiguration:

Testzeit	28.10.15, 14:50-15:10 Uhr, 20 Minuten
Ksplice Version	1.2.24
Kern	3.10.0-123.1.2.el7 / RCK
Oracle	*Oracle-Linux 7*
Swingbench	15 Nutzer, standard Transaktionen
Abweichungen	Deinstallation der *Ksplice*-Patches [5cqek217], [q64nnu1u], [zosl3oug], [lcvvcmyf]

Ergebnisse:

Auffälligkeiten	keine
program.log	keine Aufzeichnung
dienste.log	keine Aufzeichnung
kern.log	keine Aufzeichnung
alert_orcl.log	keine Aufzeichnung

alert_+ASM.log	keine Aufzeichnung
alert.log	keine Aufzeichnung
Swingbench	Keine Auffälligkeiten in den Logs, 10554 TPM
Datenbank / Grid	*Swingbench* startete normal, DB verarbeitete Transaktionen
Oracle-Linux	System reagiert angemessen, keine Probleme festzustellen

Auswertung:

Dieser Testlauf lief ohne Probleme durch. Leider konnten nur vier Patches deinstalliert werden. Auch der Versuch die Patches über die *Uptrack*-Konfiguration, unter `/etc/uptrack/uptrack.conf`, beim Systemstart mit zu installieren funktionierte nicht. Die Einstellungen `upgrade_on_reboot` wurde dafür einkommentiert und auf „yes“ gesetzt.

Testlauf Nr.:	**5**

Konfiguration:

Testzeit	28.10.15, 16:00-16:20 Uhr, 20 Minuten
Ksplice-Version	1.2.12
Kern	2.6.39-400.17.1.el6uek / UEK
Oracle	*Oracle-Linux 6*
Swingbench	15 Nutzer, standard Transaktionen
Abweichungen	

Ergebnisse:

Auffälligkeiten	keine
program.log	keine Aufzeichnung
dienste.log	keine Aufzeichnung
kern.log	keine Aufzeichnung
alert_orcl.log	kein ungewöhlicher Eintrag
alert_+ASM.log	kein ungewöhlicher Eintrag
alert.log	kein ungewöhlicher Eintrag
Swingbench	Keine Auffälligkeiten in den Logs, 12283 TPM
Datenbank / Grid	*Swingbench* startete normal, DB verarbeitete Transaktionen
Oracle-Linux	System reagiert angemessen, keine Probleme festzustellen

Auswertung:

Im fünften Testlauf trat kein Fehler auf, alle Patches (348) wurden installiert. Das System lief danach fehlerfrei und es wurden keine auffälligen Einträge in den Log-Dateien gefunden.

Testlauf Nr.:	**6**
Konfiguration:	
Testzeit	28.10.15, 16:30-16:50 Uhr, 20 Minuten
Ksplice-Version	1.2.12
Kern	2.6.39-400.264.4.el6uek / UEK
Oracle	*Oracle-Linux 6*
Swingbench	15 Nutzer, standard Transaktionen
Abweichungen	Deinstallation aller *Ksplice*-Patches
Ergebnisse:	
Auffälligkeiten	keine
program.log	keine Aufzeichnung
dienste.log	keine Aufzeichnung
kern.log	keine Aufzeichnung
alert_orcl.log	kein ungewöhlicher Eintrag
alert_+ASM.log	kein ungewöhlicher Eintrag
alert.log	kein ungewöhlicher Eintrag
Swingbench	Keine Auffälligkeiten in den Logs, 13043 TPM
Datenbank / Grid	*Swingbench* startete normal, DB verarbeitete Transaktionen
Oracle-Linux	System reagiert angemessen, keine Probleme festzustellen
Auswertung:	
	Auch im sechsten Testlauf trat kein Fehler auf, alle Patches konnten wieder deinstalliert werden. Das System lief danach Fehlerfrei und es wurden keine auffälligen Einträge in den Log-Dateien gefunden.

Testlauf Nr.:	**7**
Konfiguration:	
Testzeit	23.10.15, 35 Minuten
Ksplice-Version	1.2.12
Kern	2.6.32-71.el6.x86_64 / RCK
Oracle	*Oracle-Linux 6*
Swingbench	15 Nutzer, standard Transaktionen
Abweichungen	
Ergebnisse:	
Auffälligkeiten	Install [q4n7w8t6] CVE-2010-3067: Information leak in sys_io_submit.

program.log	keine Aufzeichnung
dienste.log	keine Aufzeichnung
kern.log	Oct 23 14:08:01 c-moh-ksplice-01 kernel: hrtimer: interrupt took 27953356 ns
alert_orcl.log	kein ungewöhlicher Eintrag
alert_+ASM.log	kein ungewöhlicher Eintrag
alert.log	keine Aufzeichnung
Swingbench	Keine Auffälligkeiten in den Logs. 14350 TPM
Datenbank / Grid	*Swingbench* startete normal, DB verarbeitete Transaktionen
Oracle-Linux	System reagiert angemessen, keine Probleme festzustellen

Auswertung:

In diesem Testlauf wurde die Installation der Patches unterbrochen, da ein Prozess die entsprechende Funktion gerade nutzt. Dieses mal war ein Prozess der Datenbank `- oracle (pid 3644)`. Bei diesem Testlauf musste nur ein zweites mal der Aktualisierungsvorgang gestartet werden, bis alle Patches installiert waren. Die Wartezeit zwischen dem ersten und zweiten Aktualisierungsversuch betrug 8 Minuten. In den Aufzeichnungen war kein Hinweis auf eine Instabilität des Systems zu erkennen.

Testlauf Nr.:	**8**

Konfiguration:

Testzeit	23.10.15, 20 Minuten
Ksplice-Version	1.2.12
Kern	2.6.32-573.7.1.el6.x86_64 / RCK
Oracle	*Oracle-Linux 6*
Swingbench	15 Nutzer, standard Transaktionen
Abweichungen	Deinstallation aller *Ksplice*-Patches

Ergebnisse:

Auffälligkeiten	keine
program.log	keine Aufzeichnung
dienste.log	keine Aufzeichnung
kern.log	keine Aufzeichnung
alert_orcl.log	kein ungewöhlicher Eintrag
alert_+ASM.log	kein ungewöhlicher Eintrag
alert.log	kein ungewöhlicher Eintrag
Swingbench	Keine Auffälligkeiten in den Logs, 14227 TPM
Datenbank / Grid	*Swingbench* startete normal, DB verarbeitete Transaktionen

Oracle-Linux	System reagiert angemessen, keine Probleme festzustellen

Auswertung:

Der Test lief wie geplant durch. Es wurden keine Hinweise auf eine Instabilität gefunden.

Tabelle D.1: Programm-Parameter-Übersicht

D.2 Aufgetretene Fehler

Diese Fehler traten bei der Installation / Deinstallation von *Ksplice*-Patches auf. Sie werden verursacht, wenn die aus zu tauschende Kern-Funktion gerade von einem oder mehrere Prozesse verwendet wird und daher nicht verändert werden kann.
Fehler von Testdurchlauf 1:

```
Error processing ksplice-vfi7yfgq.tar.gz
The following actions failed:
Install vfi7yfgq] Denial-of-service when exiting processes.
Ksplice was unable to install the update because one or more programs
are constantly using the kernel functions patched by this update. You
should be able to install this update by trying again. If trying
again does not work, please report this problem to <ksplice-
support_ww@oracle.com>. Even if trying again does not work, closing
the following programs should make it possible to install this update:
- ohasd.bin (pid 2120)
- ohasd.bin (pid 2119)
- ohasd.bin (pid 2118)
```

Fehler von Testdurchlauf 2:

```
Error processing ksplice-bejk62l4.tar.gz
Effective kernel version is 3.8.13-68.3.3.el7uek
The following actions failed:
Remove [bejk62l4] CVE-2015-2830:
mis-handling of int80 fork from 64bits application.
Ksplice was unable to remove the update because one or more programs
are constantly using the kernel functions patched by this update. You
should be able to install this update by trying again. If trying
again does not work, please report this problem to <ksplice-
support_ww@oracle.com>. Even if trying again does not work, closing
the following programs should make it possible to install this update:
- kworker/1:0 (pid 12502)
- kworker/1:2 (pid 12406)
- kworker/0:2 (pid 12386)
- kworker/1:1 (pid 12244)
- kworker/0:0 (pid 11736)
```

Fehler von Testdurchlauf 3:

```
Install [aafjnrui] CVE-2014-3153: Local privilege escalation in futex
requeueing. Ksplice was unable to install the update because one or
```

```
more programs are constantly using the kernel functions patched by this
update. You should be able to install this update by trying again.
If trying again does not work, please report this problem
to <ksplice-support_ww@oracle.com>. Even if trying again does not work,
closing the following programs should
make it possible to install this update:
- ocssd.bin (pid 2201)
- ocssd.bin (pid 2200)
- ocssd.bin (pid 2199)
- ocssd.bin (pid 2197)
- ocssd.bin (pid 2196)
- ocssd.bin (pid 2194)
- ocssd.bin (pid 2192)
- ocssd.bin (pid 2191)
- ocssd.bin (pid 2188)
- ocssd.bin (pid 2181)
- ocssd.bin (pid 2179)
- cssdagent (pid 17672)
- cssdagent (pid 17642)
- cssdagent (pid 2177)
- cssdagent (pid 2175)
- cssdagent (pid 2173)
- cssdagent (pid 2170)
- cssdagent (pid 2160)
- cssdagent (pid 2156)
- evmlogger.bin (pid 2126)
- evmd.bin (pid 2107)
- oraagent.bin (pid 11824)
- oraagent.bin (pid 5543)
- oraagent.bin (pid 3633)
- oraagent.bin (pid 2346)
- oraagent.bin (pid 2131)
- oraagent.bin (pid 2111)
- oraagent.bin (pid 2102)
- oraagent.bin (pid 2101)
- oraagent.bin (pid 2099)
- oraagent.bin (pid 2098)
- oraagent.bin (pid 2097)
- oraagent.bin (pid 2096)
- oraagent.bin (pid 2093)
- oraagent.bin (pid 2092)
- oraagent.bin (pid 2090)
- ohasd.bin (pid 2043)
```

```
- ohasd.bin (pid 2042)
- ohasd.bin (pid 2041)
- ohasd.bin (pid 2040)
- ohasd.bin (pid 2039)
- ohasd.bin (pid 2038)
- ohasd.bin (pid 2037)
- ohasd.bin (pid 2036)
- ohasd.bin (pid 2033)
- ohasd.bin (pid 2032)
- ohasd.bin (pid 2030)
- ohasd.bin (pid 2029)
- ohasd.bin (pid 2028)
- ohasd.bin (pid 2027)
- ohasd.bin (pid 2024)
- ohasd.bin (pid 2023)
- ohasd.bin (pid 2022)
- ohasd.bin (pid 2021)
- ohasd.bin (pid 2020)
- ohasd.bin (pid 1619)
- rs:main (pid Q:Reg)
- auditd (pid 586)
```

Fehler von Testdurchlauf 7:

```
Error processing ksplice-q4n7w8t6.tar.gz
The following actions failed:
Install [q4n7w8t6] CVE-2010-3067: Information leak in sys_io_submit.
Ksplice was unable to install the update because one or more programs.
are constantly using the kernel functions patched by this update. You.
should be able to install this update by trying again. If trying.
again does not work, please report this problem to <ksplice-
support_ww@oracle.com>. Even if trying again does not work, closing.
the following programs should make it possible to install this update:.
- oracle (pid 3644).
```

Anhang E: Oracle-Support-Protokolle

E.1 Update Datenbank Bibliotheken

SR 3-11337840291 : How stable is a oracle DB, if i updating the system-library during the runtime.

Severity	4-Minimal	Status	Closed
Escalation Status	Never Escalated	Opened	08-Sep-2015 12:07 (1+ month ago)
Last Updated	15-Oct-2015 16:13 (7 days ago)		
Bug Reference	No Related Bugs	Attachments	No Related Attachments
Related Articles	1467060.1	Related SRs	No Related SRs
Support Identifier	15683751		
Account Name	ASPICON GmbH		
Primary Contact	Michael Ohme	Alternate Contact	
System		Host	No Related Hosts
Product	Oracle Database - Enterprise Edition		
		Product Version	11.2.0.4
Operating System	Linux x86-64	OS Version	Oracle Linux 6

Problem Description

Hello@all,

i have some theoretical question!

i run a oracle database (11.2.04) on a high availability server with oel6. I update my system with yum and there is a bug-fix in a, for example openssh-library.
1. When use the oracle-database-process the new version of the lib?
2. Load the database libraries at start-time or at run-time?

If a new library version is updated and the old version of the library still in ram.
3. What library is use for a new database-process?

Is it save to update my system with yum during the runtime of a database?

Thank you a lot!
Best regards, Michael.

History

ODM Knowledge Content — **Oracle Support**- 15-Oct-2015 16:13 (7 days ago)
Knowledge Review completed.

ODM Answer — **Oracle Support**- 15-Oct-2015 16:12 (7 days ago)
Yes, Oracle processes uses the OS libraries when there is a need of the libraries but not all the time.

If any OS package updates happens when database is up it may not cause any issues unless no simultaneous functions happens (update of OS package and sharing the same package by DB)

But it is recommended to make the DB down while you update the packages at OS level not to cause any issues.

Notes — **Oracle Support**- 15-Oct-2015 16:11 (7 days ago)
Hello Michael,

Thanks for the update.

As confirmed I will be closing the SR now.

Thanks for using My Oracle Support.

Thank You,
Hari
Oracle Global Customer Services
Work Days: Mon-Fri
Work Hours(Mon-Fri): 8 AM - 4.30 PM GMT (UK) || 1.30 PM to 10.00 PM IST (India) || 12:30 AM to 09:00 AM PST (AMER)

Should you need assistance outside my working hours at any stage during the life-time of this SR, please call the Support Center telephone number listed at http://www.oracle.com/us/support/contact/index.html ,choose option 1,then option 2 and ask to speak to the next available engineer.

Close Requested By Customer — **MOH@ASPICON.DE**- 15-Oct-2015 15:02 (7 days ago)
Reason for Closing : Cust Close-solution provided
Closing Remarks : Thank you for your help

Notes — **Oracle Support**- 15-Oct-2015 14:09 (7 days ago)
Hello Michael,

Thanks for the update.

As per Oracle recommendations you may face the error as you mentioned in the last update. Thanks for confirming that back.

Could you please allow me to close the SR.

Thank You,
Hari
Oracle Global Customer Services
Work Days: Mon-Fri
Work Hours(Mon-Fri): 8 AM - 4.30 PM GMT (UK) || 1.30 PM to 10.00 PM IST (India) || 12:30 AM to 09:00 AM PST (AMER)

Should you need assistance outside my working hours at any stage during the life-time of this SR, please call the Support Center telephone number listed at http://www.oracle.com/us/support/contact/index.html ,choose option 1,then option 2 and ask to speak to the next available engineer.

Update from Customer **MOH@ASPICON.DE**- 15-Oct-2015 13:47 (7 days ago)
Hi Hari,

sorry for the long time of silence, i was on vacation and had things to do with a higher priority.
My test can confirm your statement: "Oracle processes uses the OS libraries when there is a need of the libraries but not all the time."
Here is a strace snippet:
8308 17:31:21.432335 open("/u01/app/12.1.0/grid_1/lib/librt.so.1", O_RDONLY) = -1 ENOENT (No such file or directory)
8308 17:31:21.432395 open("/etc/ORCLcluster/lib/tls/x86_64/librt.so.1", O_RDONLY) = -1 ENOENT (No such file or directory)
8308 17:31:21.433326 open("/lib64/librt.so.1", O_RDONLY) = 3
As you can see the database try to load the librt.so.1, first from ORACLE_HOME.

If i updated my system during runtime of the database. New started processes use the updated version
of a library such as to see in the "lsof" output below:
oracle 4629 oracle DEL REG 253,0 786828 /lib64/librt-2.12.so
oracle 11945 oracle mem REG 253,0 47112 786656 /lib64/librt-2.12.so
the risk of an error on the base of an interprocess communication is small.
but it is possible to fail, so you recommended to shut down the database.

Thank you for your great help!

bye Michael

Notes **Oracle Support**- 13-Oct-2015 15:21 (9 days ago)
Hi Michael,

Could you please update the SR whether your issue got addressed or need any more assistance.

Thank You,
Hari
Oracle Global Customer Services
Work Days: Mon-Fri
Work Hours(Mon-Fri): 8 AM - 4.30 PM GMT (UK) || 1.30 PM to 10.00 PM IST (India) || 12:30 AM to 09:00 AM PST (AMER)

Notes **Oracle Support**- 05-Oct-2015 11:25 (18 days ago)
Hi Michael,

Could you please update the SR whether your issue got addressed or need any more assistance.

Thank You,
Hari
Oracle Global Customer Services
Work Days: Mon-Fri
Work Hours(Mon-Fri): 8 AM - 4.30 PM GMT (UK) || 1.30 PM to 10.00 PM IST (India) || 12:30 AM to 09:00 AM PST (AMER)

Notes **Oracle Support**- 24-Sep-2015 12:32 (29 days ago)
Hi Michael,

Could you please update the SR whether your issue got addressed or need any more assistance.

Thank You,
Hari

Oracle Global Customer Services
Work Days: Mon-Fri
Work Hours(Mon-Fri): 8 AM - 4.30 PM GMT (UK) || 1.30 PM to 10.00 PM IST (India) || 12:30 AM to 09:00 AM PST (AMER)

ODM Action Plan **Oracle Support**- 21-Sep-2015 17:29 (1+ month ago)

Hi Michel,

I have reviewed your last update and checked internally regarding the query.

Yes, Oracle processes uses the OS libraries when there is a need of the libraries but not all the time.

If any OS package updates happens when database is up it may not cause any issues unless no simultaneous functions happens (update of OS package and sharing the same package by DB)

But it is recommended to make the DB down while you update the packages at OS level not to cause any issues.

Please let me know if you need any more assistance.

Thank You,
Hari
Oracle Global Customer Services
Work Days: Mon-Fri
Work Hours(Mon-Fri): 8 AM - 4.30 PM GMT (UK) || 1.30 PM to 10.00 PM IST (India) || 12:30 AM to 09:00 AM PST (AMER)

Notes **Oracle Support**- 21-Sep-2015 14:16 (1+ month ago)

Hi Michael,

Thanks for the update.

Please let me check the situation and will update you.

Thanks for your patience.

Thank You,
Hari
Oracle Global Customer Services
Work Days: Mon-Fri
Work Hours(Mon-Fri): 8 AM - 4.30 PM GMT (UK) || 1.30 PM to 10.00 PM IST (India) || 12:30 AM to 09:00 AM PST (AMER)

Update from Customer **MOH@ASPICON.DE**- 17-Sep-2015 15:19 (1+ month ago)

Hi Hari,

i try to finger out what libraries are load dynamically.
the tools "strace" and "lsof" should help me.

if i use lsof as a "ora_ckpt_orcl" process, it show me libraries like this:
/lib64/libaio.so.1.0.1
/lib64/libc-2.12.so

Here are the first libs out of the ORACLE_HOME !?

Ii want to use strace, but i don't know what is the base process. All database-processes are children of 1 (init).
Is it necessary to examine all processes ore is there a head/main process?

I read the oracle doku, about processes but did not find a information to this point.
http://docs.oracle.com/cd/B10501_01/server.920/a96524/c09procs.htm

Do you know more?

Thank you!
Bye Michael

ODM Action Plan **Oracle Support**- 16-Sep-2015 13:23 (1+ month ago)

Hi Michel,

Thanks for the update.

Regarding your question, database won't use any libraries outside ORACLE_HOME.

But some cases like patching ORACLE_HOME, ORACLE Installation phase and in any relinking cases

ORACLE_HOME binaries will refer the OS libraries.

Please let me know if you need any more assistance.

Thank You,
Hari
Oracle Global Customer Services
Work Days: Mon-Fri
Work Hours(Mon-Fri): 8 AM - 4.30 PM GMT (UK) || 1.30 PM to 10.00 PM IST (India) || 12:30 AM to 09:00 AM PST (AMER)

Update from Customer **MOH@ASPICON.DE**- 16-Sep-2015 13:07 (1+ month ago)

Hi Hari,

the great goal is to test the stability if a database-server. I am not sure what to expect, this is the reason of my test ;-)!
At the moment i prepare how to test them. therefor my unusual question.

Sorry there is one more question. I hope you can help me.:
I checked the file in $ORACLE_HOME, that all fine but:
Is a oracle-database use libraries outside of the $ORACLE_HOME -Directory?
In some special cases, perhaps?

Bye Michael

Notes **Oracle Support**- 15-Sep-2015 11:33 (1+ month ago)

Hello Michael,

Thanks for the update.

Could you please let me know the expected test date. If it takes more than a week you can open a new SR when required by referring to this SR and please allow us for closure of this SR.

Thanks for using My Oracle Support.

Thank You,
Hari
Oracle Global Customer Services
Work Days: Mon-Fri
Work Hours(Mon-Fri): 8 AM - 4.30 PM GMT (UK) || 1.30 PM to 10.00 PM IST (India) || 12:30 AM to 09:00 AM PST (AMER)

Update from Customer **MOH@ASPICON.DE**- 15-Sep-2015 10:29 (1+ month ago)

Hi Hari,

thank you for your response! My test will need some time. I will send a report of my progress.

Bye Michael

ODM Action Plan **Oracle Support**- 14-Sep-2015 17:09 (1+ month ago)

Hi Michael,

Regarding your question the files residing under the directory $ORACLE_HOME/lib are *.so and *.o file

Basically *.so files are static libraries where as .o files are dynamically loaded.

Please check the *.o files under the directory $ORACLE_HOME/lib which are dynamically loaded.

Please let me know for more details.

Thank You,
Hari
Oracle Global Customer Services
Work Days: Mon-Fri
Work Hours(Mon-Fri): 8 AM - 4.30 PM GMT (UK) || 1.30 PM to 10.00 PM IST (India) || 12:30 AM to 09:00 AM PST (AMER)

Update from Customer **MOH@ASPICON.DE**- 14-Sep-2015 16:59 (1+ month ago)

Hi again,

sorry, but one question is not exactly:
"Is it possible to know what library is load dynamically? Only an example!"

I need to know what library is load to run-time a dynamic loadable library.
Not only a *.so that is load at start-time!

I need it for a test of system stability.

Thank you!

Update from Customer **MOH@ASPICON.DE**- 14-Sep-2015 16:56 (1+ month ago)

Hi Hari,

Thank you for your help!

I understand the point of relinking the "Oracle Home Binaries". I read it in the link of your last post, thank you for it!

Is it possible to know what library is load dynamically? Only an example!

Last question to this topic:
Is it possible that Ksplice update binaries that a database is use?

Bye Michael

ODM Action Plan **Oracle Support**- 14-Sep-2015 13:30 (1+ month ago)
Hi Michael,

Regarding your question we have few libraries which are dynamically loaded and few components which are statically linked.

It depends on the components. Oracle recommends to perform manual relinking of Oracle Home Binaries after OS Upgrade , Patching , Downgrade or removal of the Patch or any change which impact OS library behavior . Successful relinking shows Oracle Executable are properly linked with OS binaries.

Please let me know for more information.

Thank You,
Hari
Oracle Global Customer Services
Work Days: Mon-Fri
Work Hours(Mon-Fri): 8 AM - 4.30 PM GMT (UK) || 1.30 PM to 10.00 PM IST (India) || 12:30 AM to 09:00 AM PST (AMER)

Notes **Oracle Support**- 14-Sep-2015 11:16 (1+ month ago)
Hi Michael,

Thanks for your update.

Acknowledging the update. Will check it and let you know soon.

Thanks for patience.

Thank You,
Hari
Oracle Global Customer Services
Work Days: Mon-Fri
Work Hours(Mon-Fri): 8 AM - 4.30 PM GMT (UK) || 1.30 PM to 10.00 PM IST (India) || 12:30 AM to 09:00 AM PST (AMER)

Update from Customer **MOH@ASPICON.DE**- 14-Sep-2015 10:14 (1+ month ago)

Hi Hari,

thank you for your very helpful response. But I need more information.
Is a oracle-database use "dynamic loadable library" or load the libraries just to start time?

Use every child process the same library?
As for example following scenario:
The database start with lib A in version 1 all processes and child-processes work fine.
Next, lib A is updated to version 2. After that, a new Child process start.
Start this process now with lib A in version 2 or in version 1?

Aim some Test of oracle at this problem? If yes, is it possible to read this test protokols?

Thank you!
Michael

ODM Action Plan **Oracle Support**- 08-Sep-2015 16:13 (1+ month ago)

Hi Michael,

Library update will not do anything at the database level but some updates may require system reboot. You can check with your system admin regarding this.

If the OS get upgraded relink binaries is recommended.

Please go through the below document for reference.

Relinking Oracle Home FAQ (Frequently Asked Questions) (Doc ID 1467060.1)

Please let me know for more information.

Thank You,
Hari
Oracle Global Customer Services
Work Days: Mon-Fri
Work Hours(Mon-Fri): 8 AM - 4.30 PM GMT (UK) || 1.30 PM to 10.00 PM IST (India) || 12:30 AM to 09:00 AM PST (AMER)

Update from Customer **MOH@ASPICON.DE**- 08-Sep-2015 15:48 (1+ month ago)

Hello Hari,

thank you for your help!

I am interested to update packages, where the database have dependencies. For example system-libraries.
I can live-update the kernel with ksplice and don't need to touch the system-files. So my interest at kernel-files is more secondary.

Notes **Oracle Support**- 08-Sep-2015 13:59 (1+ month ago)

Hello Michael,

Could you please let us know whether you wanted to update the kernel version of the server using yum or installing new packages.

Thank You,
Hari
Oracle Global Customer Services
Work Days: Mon-Fri
Work Hours(Mon-Fri): 8 AM - 4.30 PM GMT (UK) || 1.30 PM to 10.00 PM IST (India) || 12:30 AM to 09:00 AM PST (AMER)

Notes **Oracle Support**- 08-Sep-2015 12:39 (1+ month ago)

Hi Michael,

Your issue is assigned to me. I am Hari from Database Install and Upgrade Team and will be glad to help you with this SR.

I am currently reviewing/researching the situation and will update the Service Request (SR) or call you as soon as we have relevant information. Thank you for your patience.

My standard work week is Mon-Fri and my work hours are 8:00 AM to 4:30 PM GMT. Should you need assistance outside my working hours at any stage during the life-time of this SR, please call the Support Center telephone number listed at http://www.oracle.com/us/support/contact/index.html ,choose option 1,then option 2 and ask to speak to the next available engineer.

Thank You,
Hari
Oracle Global Customer Services
Work Days: Mon-Fri
Work Hours(Mon-Fri): 8 AM - 4.30 PM GMT (UK) || 1.30 PM to 10.00 PM IST (India) || 12:30 AM to 09:00 AM PST (AMER)

ODM Question **Oracle Support**- 08-Sep-2015 12:26 (1+ month ago)

i have some theoretical question!

i run a oracle database (11.2.04) on a high availability server with oel6. I update my system with yum and there is

a bug-fix in a, for example openssh-library.
1. When use the oracle-database-process the new version of the lib?

2. Load the database libraries at start-time or at run-time?

If a new library version is updated and the old version of the library still in ram.
3. What library is use for a new database-process?

Is it save to update my system with yum during the runtime of a database?

Customer Problem Description **MOH@ASPICON.DE**- 08-Sep-2015 12:07 (1+ month ago)

Customer Problem Description

Problem Summary

How stable is a oracle DB, if i updating the system-library during the runtime.

Problem Description

Hello@all,

i have some theoretical question!

i run a oracle database (11.2.04) on a high availability server with oel6. I update my system with yum and there is a bug-fix in a, for example openssh-library.
1. When use the oracle-database-process the new version of the lib?

2. Load the database libraries at start-time or at run-time?

If a new library version is updated and the old version of the library still in ram.
3. What library is use for a new database-process?

Is it save to update my system with yum during the runtime of a database?

Thank you a lot!
Best regards, Michael.

Error Codes

Problem Category/Subcategory

Database Installation, Upgrade & Downgrade (Single instance Only)/Startup/Shutdown (excluding recovery issues/errors)

Uploaded Files

Interactive Solution Details

Question: Choose the problem area related to database startup and shutdown.
Answer: Database startup and related issues

Question: Choose the topic that requires assistance.
Answer: Various phases of database startup

Question: Choose the topic for further reading.
Answer: Nomount phase

Diagnosis: Review the following document to troubleshoot this issue:

Read Note:1505155.1#aref_section23 Master Note: Overview of Database Startup and Shutdown

If the above solution does not resolve your issue, proceed with the next step and submit the SR.

E.2 ASM-Kernel-Driver

SR 3-11548073981 : Are fixes of the ASM-Kernel-Module included in Ksplice?

Severity	4-Minimal	Status	Closed
Escalation Status	Never Escalated	Opened	16-Oct-2015 09:58 (7 days ago)
Last Updated	21-Oct-2015 11:29 (Wednesday)		
Bug Reference	No Related Bugs	Attachments	No Related Attachments
Related Articles	1578579.1	Related SRs	No Related SRs
Support Identifier	15683751		
Account Name	ASPICON GmbH		
Primary Contact	Michael Ohme	Alternate Contact	
System		Host	No Related Hosts
Product	Linux OS		
		Product Version	Oracle Linux 6.1
Operating System	x86_64	OS Version	

Problem Description

Hello,

i have some theoretical question!

The ASM-Kernel-Module is included in the UEK, is it updated by ksplice and normal kernel updates?
Is it the same for the redhat compatible kernel?

In which source are ASM-Kernel-Module updates or patches included?

Thank you very much!

Best regards Michael

History

ODM Knowledge Content — **Oracle Support**- 21-Oct-2015 11:29 (Wednesday)
Knowledge Review completed.

Close Requested By Customer — **MOH@ASPICON.DE**- 19-Oct-2015 16:21 (Monday)
Reason for Closing : Cust Close-other
Closing Remarks : Question is answered.

Update from Customer — **MOH@ASPICON.DE**- 19-Oct-2015 16:19 (Monday)
Hello,

thank you for the response!

Bye Michael.

Notes — **Oracle Support**- 19-Oct-2015 14:13 (Monday)
Reviewed DOC ID 1578579.1 and 1089399.1

Notes — **Oracle Support**- 19-Oct-2015 14:10 (Monday)
Hello,

Please find the answeres for your question updated below.

The oracleasm kernel driver is built into the Unbreakable Enterprise Kernel and does not need to be installed/updated manually

The oracleasm kernel driver for the Red Hat Compatible Kernel needs to be installed/updated manually from ULN

Thanks,
Prasad M

ODM Answer — **Oracle Support**- 19-Oct-2015 14:09 (Monday)
The oracleasm kernel driver is built into the Unbreakable Enterprise Kernel and does not need to be installed/updated manually

The oracleasm kernel driver for the Red Hat Compatible Kernel needs to be installed/updated manually from ULN

ODM Question — **Oracle Support**- 19-Oct-2015 14:07 (Monday)

The ASM-Kernel-Module is included in the UEK, is it updated by ksplice and normal kernel updates?
Is it the same for the redhat compatible kernel?

Customer Problem Description **MOH@ASPICON.DE**- 16-Oct-2015 09:58 (7 days ago)

Customer Problem Description

Problem Summary

Are fixes of the ASM-Kernel-Module included in Ksplice?

Problem Description

Hello,

i have some theoretical question!

The ASM-Kernel-Module is included in the UEK, is it updated by ksplice and normal kernel updates?
Is it the same for the redhat compatible kernel?

In which source are ASM-Kernel-Module updates or patches included?

Thank you very much!

Best regards Michael

Error Codes

Problem Category/Subcategory

Database Target Discovery & Monitoring - Status, Metrics, Compliance, Jobs, Reports/Status

Uploaded Files

Interactive Solution Details

Question: Choose a target type
Answer: ASM

Question: Choose an option
Answer: Cluster ASM

Question: Does one or more of the associated ASM instances show the correct status?
Answer: Yes, one or more instances are showing the correct status

Diagnosis: Review the following document to troubleshoot this issue:

Read Note:1992791.1#ClusterASM Troubleshooting an Incorrect Status issue for an ASM or Cluster ASM target in Enterprise Manager Cloud Control 12c

If the above solution does not resolve your issue, follow the steps detailed in the document below which shows how to collect the relevant information necessary for SR resolution:

Follow Note:2000681.1 SRDC - Collect information about the Enterprise Manager 12c Internal Repository Jobs responsible for calculating status availability for repository managed targets

E.3 Ksplice wird von Kworker-Prozessen blockiert

SR 3-11588994321 : Can not uninstall an ksplice-patch

Severity	3-Standard	Status	Close Requested
Escalation Status	Never Escalated	Opened	23-Oct-2015 12:02 (Friday)
Last Updated	28-Oct-2015 10:20 (6 mins ago)		
Bug Reference	No Related Bugs	Attachments	No Related Attachments
Related Articles	No Related Articles	Related SRs	No Related SRs
Support Identifier	15683751		
Account Name	ASPICON GmbH		
Primary Contact	Michael Ohme	Alternate Contact	
System		Host	No Related Hosts
Product	Private Cloud Appliance		
		Product Version	2.0.2
Operating System	Linux x86-64	OS Version	Oracle Linux 7

Problem Description

Hello,

i am testing ksplice and want to delete all patches of my kernel (3.8.13-35.3.2.el7uek not-patched!)
If the system tried to delete the patch [bejk62l4] i got the following error:
The following actions failed:

Remove [bejk62l4] CVE-2015-2830: mis-handling of int80 fork from 64bits application.

Ksplice was unable to remove the update because one or more programs
are constantly using the kernel functions patched by this update. You
should be able to install this update by trying again. If trying
again does not work, please report this problem to <ksplice-
support_ww@oracle.com>. Even if trying again does not work, closing
the following programs should make it possible to install this update:

- kworker/0:2 (pid 11834)
- kworker/1:1 (pid 11691)
- kworker/1:2 (pid 11622)
- kworker/0:0 (pid 11586)
- kworker/1:0 (pid 10923)
- kworker/0:1 (pid 7751)

I tried:
- kill the processes -> autorestart of the processes -> no effect
- waited 17 houers -> same error -> no effect

Questions:
- when is it possible to delete the patch?
- where uptrack/ksplice write log-files, is it possible to configurate it in rsyslog.conf?
- i tried to delete a other single patch (uptrack-remove vjy57qbi) , but uptrack want to delete much more. have the patches (inter-)dependencys?

The System:
Virtuelle Maschine VMX-10 VMware 5.5
CPU: 2x vCPU
RAM: 4 GByte
Oracle Linux 7
Oracle-Datenbank 12c Enterprise Edition Release 12.1.0.2.0 - 64bit with ASM Grid

History

Close Requested By Customer — **MOH@ASPICON.DE** - 28-Oct-2015 10:20 (6 mins ago)

Reason for Closing : Cust Close-solution provided
Closing Remarks : Hello smruti,

I disabled the install_on_reboot flag and rebooted the System.
thank you for your help!

Best regards, Michael.

ODM Action Plan — **Oracle Support** - 28-Oct-2015 01:27 (8+ hours ago)

Hi Michael,

Ksplice has safety checks to ensure that removing an update is safe. And at the state we are in now, ksplice does not think removing the package we are trying to uninstall is safe.

>> Is it possible to install or uninstall all ksplice-patchs but skip one patch or a set of patchs?

No, it is not possible to selectively uninstall updates,

In this case you would need to reboot the system and ensure that 'install_on_reboot = no' is set in /etc/uptrack/uptrack.conf so that they are not reinstalled on reboot.

Regards,
smruti

Update from Customer **MOH@ASPICON.DE**- 27-Oct-2015 14:36 (19+ hours ago)
Hi smruti,

only as an update:
I tried a reboot, but no success, the uninstall of the patch interrupt at the same position.
I tried to install all patches and remove all again, same error.

The error shows minimum one process called "kworker".
It is not so easy to find the source process of the work-task from the workqueues.
The following link show some solution for "kworker" problems under point "7. Debugging", but this did not help me.
http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/tree/Documentation/workqueue.txt?id=HEAD

Thank you!
Best regards Michael.

Update from Customer **MOH@ASPICON.DE**- 27-Oct-2015 09:35 (Tuesday)
Hi smruti,

>Did you try "uptrack-remove" command?
Yes, i used uptrack-remove --all.

>What command did you run to apply the patch? Was it by running a complete patch?
>#uptrack-upgrade -y
The installation was fine, but the uninstall not work.

>Did you try using uptrack-show to confirm and uptrack-remove the delete patches?
Yes i only use the command line tool uptrack!
I also tried: #uptrack-remove bejk62l4, but with the same error.
Is it possible to install or uninstall all ksplice-patchs but skip one patch or a set of patchs?

>Also if possible, reboot the box with so the patch will go away.
>The above will work, if you have following parameters as follows in uptrack.conf file:
No this is not possible!

The ksplice should store its logs in yum.log and /var/log/messages.
OK THX!
The only one interesting messasge in the log files is:
Oct 23 22:00:40 c-moh-ksplice-02 kernel: hrtimer: interrupt took 14742362 ns
But this should be the normal interrupt of ksplice!?

Best regards, Michael.

ODM Action Plan **Oracle Support**- 23-Oct-2015 14:56 (Friday)
=====Oracle VM/Linux=====
Hi Michael,

Thank you for contacting us.
Did you try "uptrack-remove" command?

What command did you run to apply the patch? Was it by running a complete patch?
#uptrack-upgrade -y

Did you try using uptrack-show to confirm and uptrack-remove the delete patches?

Refer : http://www.oracle.com/technetwork/articles/linux/ksplice-update-tour-1896119.html

Also if possible, reboot the box with so the patch will go away.
The above will work, if you have following parameters as follows in uptrack.conf file:

---
install_on_reboot = no
upgrade_on_reboot = no

The ksplice should store its logs in yum.log and /var/log/messages.

Regards,
smruti

Customer Problem Description **MOH@ASPICON.DE**- 23-Oct-2015 12:02 (Friday)

Customer Problem Description

Problem Summary

Can not uninstall an ksplice-patch

Problem Description

Hello,

i am testing ksplice and want to delete all patches of my kernel (3.8.13-35.3.2.el7uek not-patched!)
If the system tried to delete the patch [bejk62l4] i got the following error:
The following actions failed:

Remove [bejk62l4] CVE-2015-2830: mis-handling of int80 fork from 64bits application.

Ksplice was unable to remove the update because one or more programs
are constantly using the kernel functions patched by this update. You
should be able to install this update by trying again. If trying
again does not work, please report this problem to <ksplice-
support_ww@oracle.com>. Even if trying again does not work, closing
the following programs should make it possible to install this update:

- kworker/0:2 (pid 11834)
- kworker/1:1 (pid 11691)
- kworker/1:2 (pid 11622)
- kworker/0:0 (pid 11586)
- kworker/1:0 (pid 10923)
- kworker/0:1 (pid 7751)

I tried:
- kill the processes -> autorestart of the processes -> no effect
- waited 17 houers -> same error -> no effect

Questions:
- when is it possible to delete the patch?
- where uptrack/ksplice write log-files, is it possible to configurate it in rsyslog.conf?
- i tried to delete a other single patch (uptrack-remove vjy57qbi) , but uptrack want to delete much more. have the patches (inter-)dependencys?

The System:
Virtuelle Maschine VMX-10 VMware 5.5
CPU: 2x vCPU
RAM: 4 GByte
Oracle Linux 7
Oracle-Datenbank 12c Enterprise Edition Release 12.1.0.2.0 - 64bit with ASM Grid

Error Codes

CVE-2015-2830

Problem Category/Subcategory

Oracle Virtual Compute Appliance Software

Uploaded Files

Template Question Responses
---
1) In addition to the Problem Description that you provided during Step 1 "General Information"
please provide any additional information that could help us to diagnose and answer your request.
sorry i did not find ksplice/uptrack or oracle-linux in the "Problem Type" filed, so i selected the nearest and wrote the software and system details in the ticket.

2) ### Describe how this problem is impacting your business. Include relevant information such as critical events (i.e., upgrade or project milestones), dates (i.e., go live dates), number of users affected, financial impact, etc.

Anhang F: Sonstige

F.1 Übersicht der Verfügbarkeitsklassen

Verfügbarkeitsklasse	Bezeichnung	Minimale Verfügbarkeit	Nichtverfügbarkeit	Ausfallzeit pro Monat	Ausfallzeit pro Jahr
VK 0	Standard-IT-System ohne Anforderungen an die Verfügbarkeit	~95%	~5%	1 Tag	Mehrer Tage
VK 1	Standard-Sicherheit nach IT-Grundschutz bei normalemVerfügbarkeitsbedarf	99,0%	1%	< 8 h	< 88 h
VK 2	Standard-Sicherheit nach IT-Grundschutz bei erhöhtem Verfügbarkeitsbedarf	99,9%	0,1%	< 44 min	< 9 h
VK 3	Hochverfügbar nach IT-Grundschutz für spezifische IT-Ressourcen	99,99%	0,01%	< 5 min	< 53 min
VK 4	Höchstverfügbarkeit	99,999%	0,001%	< 26 sec	< 6 min
VK 5	Desaster-Tolerant	max. Verfügbarkeit	0	0	0

Tabelle F.1: Übersicht der Verfügbarkeitsklassen
Übersicht der Verfügbarkeitsklassen [6]

F.2 Herkunftskategorien und Prioritäten des System-Log-Dienstes

Kategorie	Beschreibung
kern	Systemmeldungen direkt vom Kernel
auth	Meldungen vom Sicherheitsdienst des Systems (login, ...)
authpriv	Vertrauliche Meldungen der internen Sicherheitsdienste
mail	Meldungen des Mail-Systems
news	Meldungen des News-Systems
uucp	Meldungen des UUCP-Systems
lpr	Meldungen des Druckerdaemons
cron	Meldungen des Cron-Daemons
syslog	Meldungen des syslog-Daemons selbst
daemon	Meldungen aller anderer Daemon-Prozesse
user	Meldungen aus normalen Anwenderprogrammen

Tabelle F.2: Herkunftskategorien der Systemnachrichten [93]

Prioritäten	Beschreibung
emerg	Nachrichten, die kurz vor dem Systemausfall entstehen
alert	Alarmierende Nachricht, die sofortiges Eingreifen erforderlich macht
crit	Meldung über eine kritische Situation
err	Fehlermeldungen aller Art aus dem laufenden Betrieb
warn	Warnungen aller Art aus dem laufenden Betrieb
notice	Dokumentation besonders bemerkenswerter Situationen im Rahmen des normalen Betriebs
info	Protokollierung des normalen Betriebsablaufs
debug	Mitteilungen interner Programmzustände bei der Fehlersuche
none	Keine Priorität im eigentlichen Sinn, sondern dient zum Ausschluss einzelner Herkünfte

Tabelle F.3: Prioritäten von Nachrichten im Systemlog (absteigende Reihenfolge) [93]

F.3 Ksplice Patches Übersicht

[5cqek217] Clear garbage data on the kernel stack when handling signals.
[q64nnu1u] Provide an interface to freeze tasks.
[zosl3oug] CVE-2014-0196: Pseudo TTY device write buffer handling race.
[lcvvcmyf] CVE-2014-1737, CVE-2014-1738: Local privilege escalation in floppy ioctl.
[aafjnrui] CVE-2014-3153: Local privilege escalation in futex requeueing.
[k72ya70c] CVE-2014-2568: Information leak in netlink packet copying.
[bv9l4rjl] CVE-2014-2851: Integer overflow in IPv4 ping initialization.
[hrv811wz] CVE-2014-0206: Information leak in asynchronous I/O ring buffer.
[wdhfksh2] CVE-2014-4699: Privilege escalation in ptrace() RIP modification.
[p1xgg08a] CVE-2014-2672: Kernel panic in ath9k transmit.
[y7chg58r] CVE-2014-4653: Use after free in ALSA card controls.
[84pl4bi8] Use-after-free in process group scheduling when creating group.
[3bxsl1x8] Use-after-free in Common Block I/O control group.
[uncv14f4] CVE-2014-7825, CVE-2014-7826: Perf DoS and local privilege escalation.
[7gfc67yy] CVE-2014-4656: ALSA Control ID overflow.
[gtuv6gui] CVE-2014-4652: Arbitrary memory disclosure in ALSA user controls.
[meyqidyr] CVE-2014-3687: Remote denial-of-service in SCTP stack.
[jj03vh1k] CVE-2014-3673: Remote denial-of-service in SCTP stack.
[4feg8kwh] CVE-2014-3184: Invalid memory write in HID drivers.
[265f95r5] CVE-2014-3181: Memory corruption in Apple Magic Mouse USB driver.
[hk6dvrph] CVE-2014-3186: Memory corruption in PicoLCD USB driver.
[mcoqa1d4] CVE-2014-6410: Denial of service in UDF filesystem parsing.
[cbz25i1v] CVE-2014-3631: Kernel panic in keyring garbage collection.
[467qdibq] CVE-2014-3182: Invalid memory read in HID Logitech driver.
[l55orjc3] Kernel bug in network stack generic segmentation offload.
[janw2zen] CVE-2014-3940: Memory corruption during huge page migration.
[xg7hzknt] CVE-2014-8173: Denial-of-service in madvise with hugetlb support.
[m79c7o40] CVE-2014-8086: Denial-of-service on ext4 filesystem.
[4rv9qu7y] CVE-2015-1421: Privilege escalation in SCTP INIT collisions.
[2v017kt6] CVE-2014-9529: Use-after-free when garbage collecting keys.
[g4y9mdhj] CVE-2015-1593: Stack layout randomization entropy reduction.
[1cf3l1d8] CVE-2015-2830: mis-handling of int80 fork from 64bits application.
[8uj7yafj] Kernel hang on UDP flood with wrong checksums.
[v45ykx9j] CVE-2015-3636: Memory corruption when unhashing IPv4 ping sockets.
[hxn1p2to] CVE-2015-1333: Denial-of-service in the keyring subsystem.
[b6w7s8n7] CVE-2015-3212: Denial-of-service when processing SCTP ASCONF packets.

Effective kernel version is 3.10.0-123.el7

F.4 Abbildung der RPM-Headerdatei des Paketes „kile“

```
Name        : kile                         Relocations: (not relocatable)
Version     : 2.1                               Vendor: Fedora Project
Release     : 1.el6                         Build Date: Tue Sep 27 00:37:10 2011
Install Date: (not installed)               Build Host: x86-04.phx2.fedoraproject.org
Group       : Applications/Publishing       Source RPM: kile-2.1-1.el6.src.rpm
Size        : 14790237                         License: GPLv2+
Signature   : RSA/8, Tue Sep 27 12:03:36 2011, Key ID 3b49df2a0608b895
Packager    : Fedora Project
URL         : http://kile.sourceforge.net/
Summary     : (La)TeX source editor and TeX shell
Description :
Kile is a user friendly (La)TeX editor.  The main features are:
  * Compile, convert and view your document with one click.
  * Auto-completion of (La)TeX commands
  * Templates and wizards makes starting a new document very little work.
  * Easy insertion of many standard tags and symbols and the option to define
    (an arbitrary number of) user defined tags.
  * Inverse and forward search: click in the DVI viewer and jump to the
    corresponding LaTeX line in the editor, or jump from the editor to the
    corresponding page in the viewer.
  * Finding chapter or sections is very easy, Kile constructs a list of all
    the chapter etc. in your document. You can use the list to jump to the
    corresponding section.
  * Collect documents that belong together into a project.
  * Easy insertion of citations and references when using projects.
  * Advanced editing commands.
```

Abbildung F.1: *RPM*-Headerdatei des Paketes „kile“

F.5 Vergleich der Nutzeranzahl von Swingbench

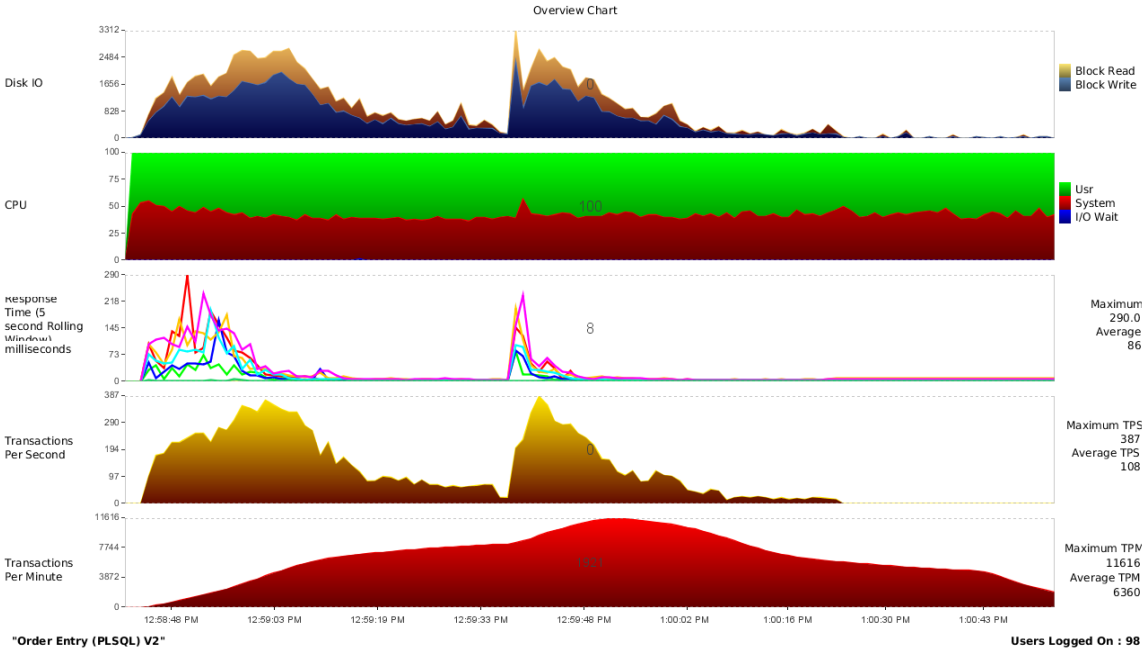

Abbildung F.2: Leistungsgraphen von *Swingbench* bei 100 Nutzer

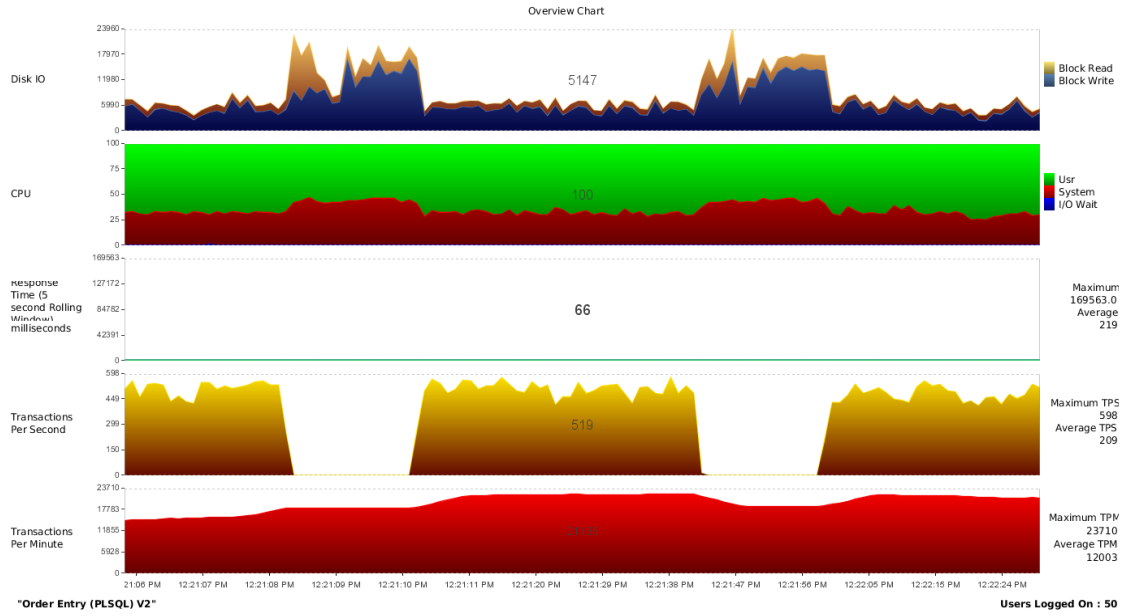

Abbildung F.3: Leistungsgraphen von *Swingbench* bei 50 Nutzer

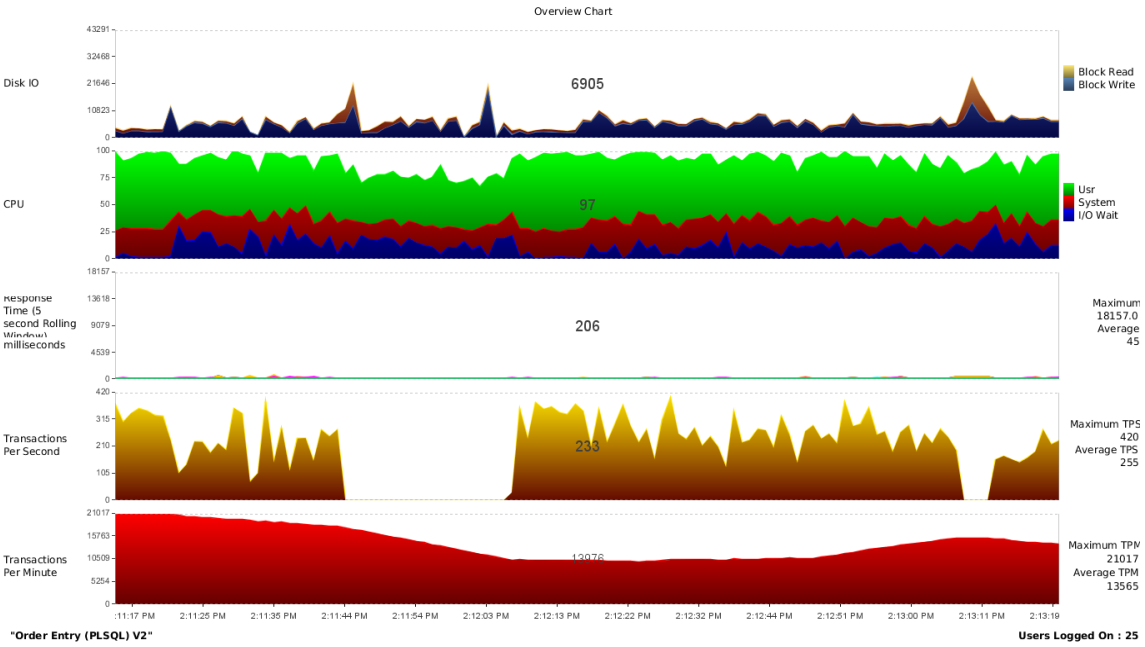

Abbildung F.4: Leistungsgraphen von *Swingbench* bei 25 Nutzer

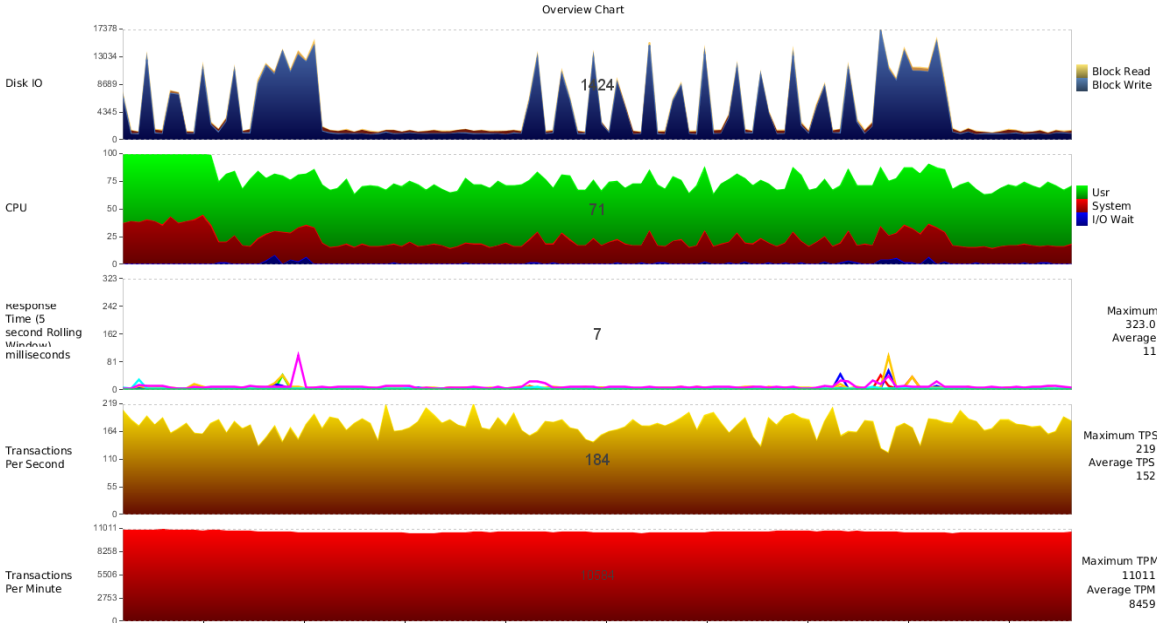

Abbildung F.5: Leistungsgraphen von *Swingbench* bei 10 Nutzer

Erklärung

Hiermit erkläre ich, dass ich meine Arbeit selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die Arbeit noch nicht anderweitig für Prüfungszwecke vorgelegt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Mittweida, 30.11.15